UMONS - Université de Mons
Faculty of Science
Computer Science Department

Synthesis in Multi-Criteria Quantitative Games

Mickaël Randour

A dissertation submitted in fulfillment of the requirements of
the degree of *Docteur en Sciences*

Advisors
Pr. Véronique Bruyère
Université de Mons, Belgium
Pr. Jean-François Raskin
Université Libre de Bruxelles, Belgium

Jury
Pr. Patricia Bouyer-Decitre
LSV - ENS Cachan, France
Pr. Thomas Brihaye
Université de Mons, Belgium
Pr. Véronique Bruyère
Université de Mons, Belgium
Pr. Thomas A. Henzinger
IST Austria, Austria
Pr. Joost-Pieter Katoen
RWTH Aachen, Germany
Pr. Hadrien Mélot
Université de Mons, Belgium
Pr. Jean-François Raskin
Université Libre de Bruxelles, Belgium

April 2014

Acknowledgments

Good acknowledgments require balance and subtlety. You do not want to get too emotional, but you still want to be true to your feelings and express your gratitude to all those individuals who helped you, somehow, someday, someplace, and ultimately supported you in completing this piece of work. People who know me will not be amazed that I chose the fully subtle way, and... Well, who am I kidding? I am *this* grateful, and I should say it!

First and foremost, I want to thank my advisors, Véronique Bruyère and Jean-François Raskin, for their untrammeled support. Véronique, you talked me into research and you believed in me back when I was a student. Jean-François, you accepted me in your team and taught me so much about this path. Both, you respected my independance and my vision. And of course, you showed me how to improve this vision more than once. Last but not least, you are terrific people, and working with you is a pleasure that I intend to pursue.

I am much obliged to the other members of my jury: Patricia Bouyer-Decitre, Thomas Brihaye, Thomas A. Henzinger, Joost-Pieter Katoen and Hadrien Mélot. I know all of you have really busy lives, and I am very happy that you accepted to review my work. I am sure your feedback and advices will be useful.

Thomas, you really deserve your own paragraph. I remember my first course with you when I was still young - just before I went for the hippie hair. Since then you have always been kind of a safe haven for me. Always a nice thought, a pat on the back, and most importantly, a laugh. I do not know if I would have finished this without you, but what I do know for sure is that it would have been damn less fun!

I want to thank my co-authors, Krishnendu Chatterjee, Laurent Doyen and

Emmanuel Filiot. Working with you was inspiring and enlightening. Krish, I would like to thank you again for your welcome in IST Austria, and the good time I spent there with Laurent and you. Manu, I am sure that there are more bets you can lose to me, so many thanks for that!

I express my thanks to all scientists in the community that I have had the luck to meet in conferences and research events during my thesis. Your feedback was quite precious. I would like to mention particularly Aaron Bohy, Guy Latouche, Guy Louchard, David Sbabo, as well as anonymous reviewers, for useful pointers and fruitful discussions.

On the same note, I am very grateful to my colleagues, both in UMONS and ULB, for their great company and enthralling discussions. I would also like to thank these institutions as a whole, along with the F.R.S.-FNRS, for their financial support through a research fellowship.

Now I will switch to French as I say a word about people that made my life brighter, digressing from the strictly scientific context.

Je remercie tous les professeurs qui ont participé à mon éducation. Ici bien sûr, à l'UMH à mon arrivée et ensuite à l'UMONS : j'ai eu la chance d'en croiser beaucoup, aux départements de Physique, de Mathématique et d'Informatique. Tous ont entretenu et même développé la passion qui m'a mené jusqu'ici. Je tiens particulièrement à remercier Véronique, Alain, Bruno, Hadrien, Jef, Olivier et Tom, pour leur accueil et leur soutien. Bien entendu, je ne peux modérer mes louanges envers notre merveilleux Doyen. Christian, merci pour tes anecdotes dont la truculence n'a d'égale que le volume sonore! Je tiens aussi à remercier les secrétaires de la Faculté des Sciences pour leur aide précieuse et leur sympathie. Merci à Angélique, Claudia, Lyane, Marie-Anne et Virginie.

La place me manque pour évoquer tous mes fabuleux collègues. Merci à Aaron pour la survie californienne, à Romu pour m'avoir initié à la théorie du ping-pong, à Youssouf pour avoir supporté deux crises importantes (la rédaction de cette thèse et la pénurie de café), à Romain pour continuer à répondre au téléphone, à Dany pour les verres organisés ou impromptus, à McFly pour la *soldier attitude*, à Noémie pour nous sauver de la pénurie de café, à Jésus qui me fait rire et me passionne quelque soit le sujet de discussion (cette phrase va faire débat !), à Marc pour le *Veggie Day* et les vélos, à Julie pour les gâteaux et les promenades dans les quartiers à climat tropical, à Quentin parce qu'ainsi

chacun se demandera duquel je parle, à Fabien pour la préparation du café (le nerf de la guerre), à Cédric pour ses cheveux poivre et sel, à Rivaldo pour son sourire et sa bonne humeur... Merci à tous les autres qui ne sont pas cités mais n'en sont pas moins géniaux.

J'ai mentionné l'importance de mes professeurs d'université. J'aimerais aussi rendre hommage à celles et ceux qui ont marqué ma jeunesse: Alain, Françoise, Martine et Nicole, entre autres. *Non scholae, sed vitae discimus.*

J'ai la chance d'être entouré d'amis formidables. Mille mercis à Jéjé pour la discussion au sommet du toboggan, à Pépère pour me laisser la victoire au squash, à Greg pour les rudiments de scoutisme, à Alba pour les protéines, à Chris pour rigoler à mes blagues *et* m'en inspirer d'autres, aux Sophie pour m'économiser une ligne grâce à leur prénom partagé, à Laura, Melissa et Sindy pour supporter les zouaves en question, à John pour sa forme olympique en cours les lendemains de veille (et bien sûr pour sa couette), à Nico pour m'avoir initié aux cheveux longs (finalement, je ne sais pas si je dois dire merci), à Soucré même si je bois mon café sans sucre, à Kyne pour avoir été la moitié d'un duo mythique pendant cinq ans, à Sa Majesté car il est chef de village, et à Jérôme pour sa cruauté légendaire. Merci encore à tous ceux que je ne peux citer ici...

Un grand merci à ma famille et ma belle-famille pour leur patience et leurs tentatives de compréhension de mes recherches (dédicace spéciale à Thierry !). Une pensée particulière pour mes parents : merci pour votre présence et votre soutien, tout au long de ma vie.

Finalement, j'aimerais remercier les trois personnes qui ont supporté mes absences répétées, et qui m'ont soutenu depuis toujours. Merci Cassy, Joshua et Lily pour tout le bonheur que vous m'apportez.

Abstract

Verification and synthesis are successful applications of computer science, based on extensive formal bases. Model checking is supported by powerful tool suites and is now an essential part of many high-end industrial processes. Synthesis is promising and should have important long-term impact.

One of the crucial changes over the last decade is the evolution from *Boolean* to *quantitative specifications*, giving more expressive models that describe quality of service or performance of systems. However, current models cannot account for *trade-offs* and *interplays* between quantitative aspects, which naturally arise in many applications. This thesis participates in the shift from *single-criterion quantitative models* toward *multi-criteria ones*. Our study is focused on the game-theoretic framework, modeling the interactions between a reactive system and its environment as a competitive two-player game.

Our contributions are of two kinds. On the one hand, we obtain *new results on existing models* and *improve their tractability*. On the other hand, we introduce *novel models* with interesting properties. Questions we address include deciding the winner in games with rich winning objectives, establishing efficient synthesis algorithms, bounding the memory needs for strategies, and other related problems.

We cover three axes of research. First, we study games with *multi-dimension quantitative objectives*. We prove surprising complexity results for the total-payoff and establish an optimal synthesis algorithm for mean-payoff and energy objectives along with a parity condition. Second, we introduce *window objectives*, which provide a framework to reason about quantitative behaviors in time frames. Those new objectives also approximate classical ones while avoiding a

long-standing complexity barrier. Finally, we develop the novel concept of *beyond worst-case synthesis*, combining worst-case and expected value requirements for synthesized system controllers. We study it for two important quantitative settings: mean-payoff and shortest path. For the former, we show that it provides additional expressive power for free complexity-wise.

Keywords: formal methods, computer-aided verification and synthesis, game theory, quantitative models, stochastic models, weighted games, Markov decision processes, multi-dimension games, window games, beyond worst-case synthesis, complex systems

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Verification and Synthesis via Game Theory ◇ Goals and Contributions ◇ Other Related Work

Reactive computer systems bear inherent complexity due to continuous interaction with their *environment*. While their correctness is often crucial, the design of such safety-critical systems is notoriously hard.

Formal verification and synthesis have proved to be successful applications of computer science, supporting the construction of provably safe system controllers. Many techniques take roots in the *game-theoretic framework*, modeling the interactions between the system and its environment as a competitive game.

We open this thesis with a brief presentation of the core concepts of the research field, partly inspired by [Ran13]. Afterwards, we discuss the main direction of our work, which is the study of *multi-criteria quantitative models* in a broad sense. We summarize our most important contributions. Finally, we close the chapter by mentioning other related work, exploring complementary areas of the field.

1.1 Verification and Synthesis via Game Theory

Computers are like Old Testament gods; lots of rules and no mercy.

Joseph Campbell, *The Power of Myth*.

It is no wonder that this judgment was enacted by a man renowned for his work on *myths*. Indeed, a common myth of the Digital Age is the one of *foolproof computer systems*. The quote says it all: computers are rigid systems, bent to the mathematical rules of their programs. And they have no mercy: if those programs are defective, then failures are bound to happen, with potentially unforeseen consequences.

This thesis participates in a global research effort to provide formal bases and tools to detect, avoid and prevent such failures. We present the general context in the following pages.

Reactive systems. *Reactive systems*, as named by Harel and Pnueli [HP85], are computer systems that continuously interact with their *environment*. They are now ubiquitous and their correctness is often critical. Think about control programs for power plants, ABS for cars, railway traffic control systems, etc. In general, the environment of those systems cannot be controlled. Therefore, we are in dire need of *system controllers* capable of sustaining a safe behavior of the system despite the potentially adversarial effects of the environment.

Formal methods. The construction of safe and efficient controllers for reactive systems is difficult and classical development techniques based on testing and prototyping are inadequate. Good developers know that testing do not capture the whole picture: never will it *proves* that no flaw is present in the considered system. So for critical systems, so-called *formal methods* are essential to assert the correctness of system controllers. That means using mathematical tools to prove that the system follows a given *specification* modeling desired behaviors.

To improve the design and verification of system controllers, model checking techniques were introduced in the 80s through the work of Clarke, Emerson, Vardi, Wolper, and others [CE81, VW86, CGP00]. A comprehensive book on the subject was recently authored by Baier and Katoen [BK08]. Most techniques are

based on logic and automata, which became popular thanks to seminal works by Büchi, Rabin, etc [Büc60, Rab69]. Model checking has proved relevant in the high-tech industry (e.g., Intel, IBM, Airbus). Model checking is a *verification* method: it applies *a posteriori*, checking that a preexisting formal model of a controller satisfies a given specification.

However, it is most of the time desirable to start *from* the specification and *automatically* design a controller from it, in such a way that desired properties are proved to be maintained in the process, no matter how the environment behaves. This is the more ambitious (and considerably harder) *synthesis problem*, already envisioned in the context of circuits by Alonzo Church [Chu57] and transposed in the context of reactive systems by Ramadge and Wonham [RW87], or Pnueli and Rosner [PR89], on which an important part of the research community is currently focusing its effort.

Game theory. *Game theory* provides well-suited foundations for this task. It is a wide field with extensive formal bases and applications in numerous disciplines as diverse as economics, biology, operations research and, of course, computer science. Games model interactions between cooperating and/or competiting players who play to the best of their abilities in order to satisfy individual or common objectives. While interesting works of Borel [BV38], Zermelo [Zer13] and even Cournot [Cou38] precede them, von Neumann and Morgenstern are generally considered as the "Founding Fathers of (Modern) Game Theory" through their 1944 book entitled *Theory of Games and Economic Behavior* [vNM44]. Another seminal book on the subject was authored by Osborne and Rubinstein [OR94].

The synthesis problem as a game. In this thesis, we focus on *two-player games played on graphs*. See for example work by Thomas et al. [Tho95, GTW02]. One player represents the controller and its opponent represents the environment. Players alternatively decide how to move a pebble on a graph where vertices model states of the system to control. Their moves represent actions over the system. They choose how to move the pebble according to *strategies*, creating an infinite sequence of states called play that represents the behavior of the system. The goal of the controller is to enforce a given specification, encoded through a *winning objective*: a play is winning if it belongs to a predefined set of acceptable plays. The goal of the environment, considered antagonistic, is to prevent the controller from enforcing its specification.

Throughout this thesis, we focus on the game-theoretic formalism: we do not continuously refer to the system controller and its environment but instead to player 1 ($\mathcal{P}_1$) and player 2 ($\mathcal{P}_2$).

In general, we are looking for *winning strategies* for $\mathcal{P}_1$: strategies ensuring that the outcome of the game will be a winning play, no matter what is the strategy followed by $\mathcal{P}_2$. In our context, establishing winning strategies for $\mathcal{P}_1$ corresponds to synthesizing implementable models of provably correct controllers. The synthesis process is depicted in Fig. 1.1.

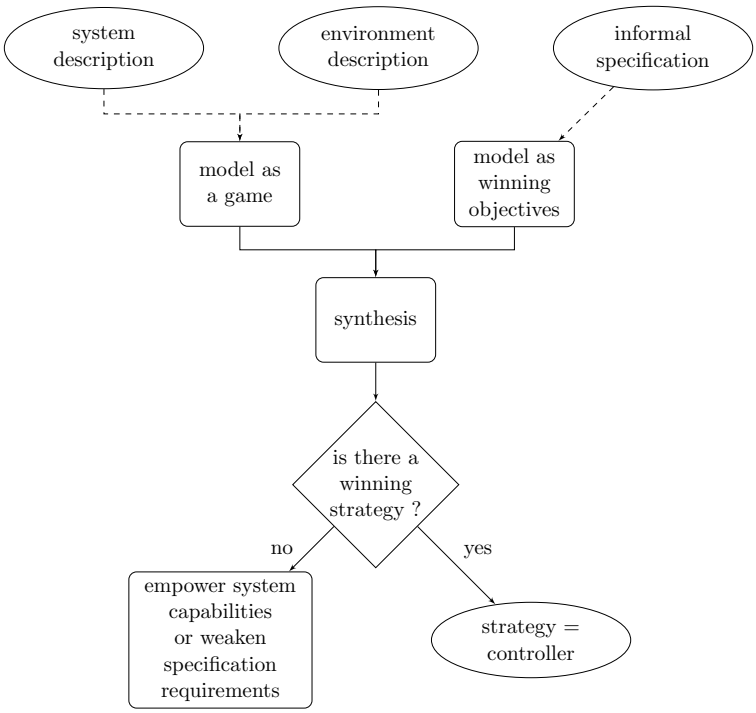


Figure 1.1: Controller synthesis through game theory: process.

Such game-theoretic formulations have proved useful in the context of synthesis [Chu62, RW87, PR89], verification [AHK02], refinement [HKR02], and compatibility checking [dAH01] of reactive systems, as well as to obtain algorithms for checking emptiness of tree automata [Tho97].

Decision problems and strategy synthesis. The focus of our work is mostly of *fundamental nature*: we investigate the complexity and expressiveness of sev-

eral classes of games and objectives. The three classical questions, given a game, an objective and an initial state of the game, are *whether one of the player can guarantee victory*, and if so, *which one* and *how*?

Interestingly, very general results by Donald A. Martin guarantee *determinacy* for most game models useful in verification and synthesis [Mar75, Mar98]: that is, for any initial state, either $\mathcal{P}_1$ can ensure a victory, or $\mathcal{P}_2$ can ensure a victory. In particular, the objectives studied in this thesis define Borel sets, hence Martin's theorems apply and guarantee determinacy. This solves the first question.

Now for the second question: can we decide who wins a game and how computationally hard is it to decide it? This decision is sometimes referred to as *solving* the game. It is a central question for us: we want to assess if $\mathcal{P}_1$ has a winning strategy (i.e., if there exists a correct controller) or not. In practice, *decidability* of the problem is not enough. It also needs to be *tractable* to preserve practical applicability of the models. Precise complexity classes are investigated, and large effort is put in *efficient algorithms and data structures*. In the following, we assume the reader to be familiar with basic notions of computational complexity. If not the case, we suggest comprehensive books by Papadimitriou [Pap94] or Sipser [Sip97].

Lastly, assuming there exists a winning strategy for $\mathcal{P}_1$, it is crucial to determine *how complex* such a strategy needs to be and to be able to actually synthesize it. The realm of strategies is vast: they may use *memory* (i.e., depend on the past), whether finite or infinite; they may prescribe actions *deterministically* or *randomly*, etc. Characterization of strategies requirements is of particular importance as the synthesis process aims at implementing controllers from those strategies. We favor simple strategies whenever possible: the corresponding controllers will be cheaper to produce and easier to maintain.

Qualitative specifications. Traditionally, *ω-regular objectives* have been studied [GTW02]. In this Boolean setup, a play is either correct or incorrect, with no interpretation of how well it behaves.

Let us mention some popular Boolean - or *qualitative* - objectives (a broader presentation is available in Sect. 2.2.1). The *reachability* objective asks to visit (at least once along a play) a given target set of states. Reachability games can be solved in linear time using *attractors* and they are P-complete [Bee80, Imm81].

The *Büchi objective* requires infinitely repeated visits of a target set. Its complexity has attracted much effort for a long-time [EJ91, McN93, Zie98] and an algorithm quadratic in the number of states has been recently introduced by Chatterjee and (M.) Henzinger [CH12]. The problem is also P-complete as it subsumes reachability games.

In *parity games*, states are mapped to priorities (natural numbers) and the objective requires that the least priority seen infinitely often along a play is even. This is a canonical way to encode ω-regular specifications [Tho97]. Parity games belong to the intriguing class of problems which are in NP $\cap$ coNP but not known to be in P [EJ88, EJS93, Zie98, KKV01]. The complexity class can be refined to UP $\cap$ coUP (subclass of NP $\cap$ coNP restricted to unambiguous Turing machines [Pap94]), as presented by Jurdziński in [Jur98]. Parity games also subsume modal μ-calculus model checking [EJS93].

In all the above objectives, pure memoryless strategies (i.e., using no memory and choosing actions deterministically) are known to suffice for both players.

Quantitative specifications. While qualitative specifications are sufficient to model yes-no properties (e.g., avoiding deadlocks), research effort has recently focused on *quantitative extensions* to model resource constraints such as power consumption, response time or buffer size [CdAHS03, BCHJ09]. To that end, games are played on *weighted graphs* and quantitative measures map plays to *payoffs* in the numerical domain, ranking their performance.

In that setting, the traditional decision problem is, given a game, a payoff function, a worst-case threshold in the numerical domain, and an initial state, whether $\mathcal{P}_1$ has a strategy ensuring a payoff at least equal to the worst-case threshold against any strategy of its adversary. This is called the *worst-case threshold problem*. We briefly mention several classical quantitative objectives. Extended discussion of the corresponding results is presented in Sect. 2.3.2.

One-dimension weights were considered for a variety of quantitative winning objectives such as mean-payoff [Gil57, LL69, EM79, GKK88, ZP96, Jur98, Pis99, LP07, BV07, BCD+11], total-payoff [GZ04, GS09] or energy [CdAHS03, BFL+08, CHKN12]. In all those settings, pure memoryless strategies exist for both players and the decision problem is in UP $\cap$ coUP, with no known polynomial-time algorithm.

Progress has been made to lift this setting to *multi-dimension weights* (modeling several quantitative aspects) for the mean-payoff and the energy objectives [CDHR10, VR11, VCD+12]. This shows considerably added complexity, reflected by the coNP-completeness of deciding the winner and the need for infinite-memory strategies for $\mathcal{P}_1$ in the mean-payoff case.

Conjunction between parity and one-dimension quantitative objectives has been studied in the literature: mean-payoff parity [CHJ05, BMOU11] and energy parity [CD12]. Again, both are in NP $\cap$ coNP. However, exponential memory suffices in energy parity games whereas mean-payoff parity games require infinite memory for $\mathcal{P}_1$.

Some *novel quantitative measures* have been recently introduced in order to represent more accurately important aspects of real-world reactive systems, or to provide increased tractability. For example, two complementary models of energy consumption were studied. In [BHR14], Boker et al. define battery transition systems, a discretization of a continuous battery model. In [BCKN12], Brázdil et al. present consumption games, an alternative to energy games.

Stochastic environments. In order to synthesize adequate controllers, one first has to establish reasonable assumptions on the behavior of the environment of the system. In traditional games, the environment is seen as an *antagonistic adversary*, aiming at failure of the system or minimization of its performance. The goal of verification and synthesis techniques is thus to provide strict worst-case guarantees on the performance of the system.

Another common standpoint, used in decision making and optimization, is to assume the environment to be *purely stochastic*. This is the setting of models known as *Markov decision processes* [Ber95].

For qualitative objectives, it is natural to relax the notion of (surely) winning strategy and consider *almost-surely winning* strategies: instead of requiring all outcomes to satisfy the specification, we ask that the specification is satisfied with probability one against the stochastic adversary. Markov decision processes with reachability objectives [CJH03, BK08], Büchi objectives [CH12], and parity objectives [CY90, dA97, CJH04] can all be solved in polynomial time. In all cases, memoryless strategies are sufficient.

In Markov decision processes with quantitative objectives, the classical question is to optimize some overall expected performance, without guarantee on

individual plays. The corresponding decision problem, named *expected value threshold problem*, was notably studied for the mean-payoff [Put94,FV97,Gim07], the total-payoff [FV97] and the shortest path [BT91, dA99].

1.2 Goals and Contributions

Research objective. Prevalent models permit to consider different quantitative aspects separately but not to look at their *interplay* and the resulting *trade-offs*. When studying resource usage and performance output, it is often unavoidable to account for such interplays. Indeed, achieving a given performance level on some aspect often comes at a cost on other ones (e.g., decreasing the response time may require additional computing power and energy consumption).

Our thesis participates in a shift from *single-criterion quantitative models* to *multi-criteria* ones, developing fundamental advances in this direction. For example, we pursue the previously discussed extension of quantitative models to the multi-dimension context. This is however not the only kind of multi-criteria approach we consider: we also mix problems of different natures, such as worst-case and expected value threshold problems.

Contributions. We briefly summarize our main contributions in the following paragraphs. A more detailed discussion of our results and their relation with state-of-the-art work in the field is presented in Chap. 3.

Our contributions are of two kinds. First, we obtain some *new results on existing models* and *improve* their *tractability*. Second, we introduce *novel models* with interesting properties. Questions we address include deciding the winner in games with rich winning objectives, establishing efficient synthesis algorithms, bounding the memory needs for strategies, and other related problems.

Multi-dimension quantitative objectives. Let us recall that multi-dimension mean-payoff games require infinite memory in general. To ensure controller implementability, the relevant problem is the construction of *finite-memory* winning strategies.

Together with Krishnendu Chatterjee and Jean-François Raskin [CRR12a, CRR12b,CRR13], we study games with multi-dimension mean-payoff and energy objectives *along with a parity objective*. This combination is both novel and appealing for practical applications. For example, it can express specifications

such as "eventually granting requests while maintaining a minimal response time and keeping the energy consumption lower than some threshold".

We prove tight bounds on memory, significantly improving previous results. We establish an *optimal symbolic and incremental synthesis algorithm* that uses antichains as efficient data structures [DDHR06]. We also study when randomness can be used to replace memory in related controllers to provide conceptually simpler implementations.

We settle the case of the total-payoff in joint work with Krishnendu Chatterjee, Laurent Doyen and Jean-François Raskin [CDRR13a, CDRR13b]: we prove the undecidability of the synthesis problem in multi-dimension and highlight the *need for alternative quantitative measures*.

Timing guarantees and tractable approximations. Existing objectives are not sufficient for specifications sensitive to *timing issues*. Mean-payoff and total-payoff characterize long-run behaviors over infinite plays. A desired property for many practical problems is to provide bounds on time frames in which an acceptable behavior can be witnessed (e.g., income). To overcome this limitation, we introduce a new family of objectives, called *window objectives*, that strengthen quantitative specifications with such timing guarantees [CDRR13a, CDRR13b]. This is joint work with Krishnendu Chatterjee, Laurent Doyen and Jean-François Raskin.

Additionally to increased modeling power, those objectives show *computational efficiency*. As discussed, some tractability issues arise for the classical objectives. For the one-dimension setting, both mean-payoff and total-payoff games are known to be in UP $\cap$ coUP but whether they can be solved in deterministic polynomial time is a long-standing and important question. For the multi-dimension setting, we show that total-payoff is undecidable. We prove that window objectives are *conservative approximations* of the classical measures, with complexities breaking the previous barriers: in one-dimension, window games are computable in deterministic polynomial time, and in multi-dimension, we provide a decidable fragment with optimal algorithms.

Beyond the worst-case. In collaboration with Véronique Bruyère, Emmanuel Filiot and Jean-François Raskin, we establish a framework for the analysis of *performance trade-offs* with regard to the nature of the environment [BFRR13, BFRR14a, BFRR14b].

On the one hand, classical games involve an environment which is purely antagonistic and ask for strict guarantees. On the other hand, Markov decision processes model purely stochastic environments: the aim is then to optimize the expected payoff, with no guarantee on individual plays. Up to now, both aspects could only be considered separately. In practice, a natural wish is to achieve a reasonable trade-off between them: nice expected performance in the everyday situation while ensuring a strict (but relaxed) performance threshold even in the event of very bad (while unlikely) circumstances. This is exactly the goal of the *beyond worst-case synthesis problem*.

We study it for two important quantitative settings: mean-payoff and shortest path. In both cases, we establish synthesis algorithms with optimal complexities and provide tight memory bounds. For the mean-payoff we manage to provide additional modeling power *while maintaining the same tractability level* as the simpler worst-case setting, though the problem is conceptually much harder to tackle. This is particularly interesting as the mean-payoff is one of the most widely used quantitative measures.

Thesis outline. This thesis is divided in five parts.

- *Part I - Controller Synthesis via Game Theory*. In Chap. 2, we present the necessary background: game models, classical objectives, etc. In Chap. 3, we give a detailed overview of our main contributions. The technical details are spread over the three following parts.
- *Part II - Multi-Dimension Objectives*. In Chap. 4, we discuss state-of-the-art results on multi-dimension quantitative games. We also establish our undecidability result for multi-dimension total-payoff games. In Chap. 5, we present the memory bounds. The synthesis algorithm is developed in Chap. 6. Finally, the question of randomness instead of memory is addressed in Chap. 7.
- *Part III - Window Objectives*. We introduce window objectives and study their relation with classical objectives in Chap. 8. Our results for one-dimension games are presented in Chap. 9. In Chap. 10, we discuss the multi-dimension setting.
- *Part IV - Beyond Worst-Case Synthesis*. The beyond worst-case framework is defined in Chap. 11. We study it for the mean-payoff function in

Chap. 12 and for the shortest path in Chap. 13.

- *Part V - Discussion.* In Chap. 14, we briefly summarize our results and the remaining open questions. Then, we review several promising research directions linked to our research.

1.3 Other Related Work

We very briefly mention some complementary research directions. Extension or adaptation of our results in those contexts could be interesting questions to address. This list is obviously (and knowingly) far from complete.

Timed and hybrid systems. In classical graph games, every change is discrete. It is sometimes necessary to consider *real-time* constraints for embedded systems. Alur and Dill have introduced *timed automata* to model the continuous character of time [AD90, AD94]: they extend finite automata with real-valued clocks. This has been the start of a fruitful research topic, which has attracted focus of the community over the last years (see for example Brihaye's thesis [Bri06]). Powerful tool suites exist for the model checking of real-time systems, such as Uppaal [BDL+06].

The more general theory of *hybrid automata* was introduced by (T.A.) Henzinger to integrate arbitrary derivative of continuous variables [Hen96]. While many problems are undecidable for the general class of hybrid automata, semi-algorithms are provided in tools such as HyTech [HHWT97]. Another interesting extension of timed automata is the model of *timed games* [AMP95, AMPS98, San13].

Multi-player games. We focus on games between two players, respectively representing a system controller and the environment of the system. In some applications, it is useful to account for multiple interdependent components, with objectives that are not necessarily antagonistic. This is the focus of *multi-player non-zero-sum games*. See for example [GU08, Umm10, Bre13, De 13].

Imperfect information. Our setting is one of *perfect information*: both players know exactly what is the current state of the system at all times. In some contexts, this assumption is too strong: a system controller may only be able to obtain *partial information* about the state of the controlled system. This extension proves to be much harder to tackle, involving exponential blow-ups

due to subset constructions. For example, imperfect information parity games are EXPTIME-complete [Rei84, CDHR06]. For quantitative objectives, decidability is easily lost: mean-payoff games are undecidable in full generality and restrictions need to be introduced to regain tractable models [DDG+10, HPR13].

Concurrent games. In this thesis, we consider *turn-based* games: players choose their actions alternatively. Games where players choose actions *concurrently* have been studied for a variety of winning objectives [dAHK98, Cha07a, BBMU12, KLST12].

Probabilistic systems and stochastic games. A number of studies on the verification of *probabilistic systems* (like Markov decision processes) for multi-objective properties were recently developed [EKVY08, KP13], along with tools such as the probabilistic model checker PRISM [KNP11].

On the same note, the extension of two-player games to stochastic transition functions was studied both for qualitative objectives [Eve57, Cha07b] and for quantitative ones [Sha53, LL69, MN81, TV87, CFK+13b], as well as for conjunction of both kinds [CFK+13a, CDGO14].

Part I

Controller Synthesis via Game Theory

CHAPTER 2

Background

Games, Markov Decision Processes and Markov Chains ◇ Objectives and Decision Problems ◇ Some Classical Objectives

This chapter presents the fundamental models studied in this thesis. In games, one player is faced to an antagonistic adversary. In Markov decision processes, the adversary is stochastic. Markov chains are purely stochastic processes.

Players try to achieve winning objectives. They behave according to well-chosen strategies. Strategies are analyzed with regard to several criteria such as their worst-case payoff guarantee, their expected payoff in a stochastic environment, their memory requirements and so on.

We illustrate many usual winning objectives and review important results over the related models. We notably discuss reachability, Büchi, parity, mean-payoff, total-payoff, shortest path and energy objectives. This chapter focuses on games with a single or one-dimension objective.

2.1 Games, Markov Decision Processes and Markov Chains

All the models studied in this thesis are based on *directed graphs*. The vertices are called *states*. A pebble is placed in an initial state and moved from state to state, creating an infinite path in the graph, called *play*. How the pebble moves depends on who possesses the origin state. States can either belong to a player or be stochastic. In the first case, the player chooses where to move the pebble according to his *strategy*. In the second one, the pebble is moved according to a predefined *probability distribution* over successor states in the graph.

Players choose their actions in order to create plays that belong to a set of winning plays, named *winning objective*. We mostly focus on *two-player games with zero-sum objectives*. That is, the first player wins if he achieves some objective while the second player wins if the first player fails. Some objectives are *quantitative*: plays are mapped to numerical values. The objective of the first player is then to ensure a value higher (resp. lower) than a given threshold while the second player tries to minimize (resp. maximize) this value.

In this section, we present the basic notions that formalize these concepts. We focus on the classical framework. We refine or extend some concepts in the following chapters.

2.1.1 Weighted Graphs and Plays

Definition 2.1 (Graph). A *weighted directed graph* is a tuple $\mathcal{G} = (S, E, w)$ where (i) S is the set of vertices, called *states*; (ii) $E \subseteq S \times S$ is the set of directed edges; and (iii) $w \colon E \to \mathbb{Z}$ is the weight labeling function.

Since we only work with directed graphs in the following, we omit the adjective and talk about *weighted graphs*. Also, in the sequel, we almost exclusively work with *finite* graphs, i.e., graphs for which the set of states S is finite. Given a state $s \in S$, we denote by $\mathsf{Succ}(s) = \{s' \in S \mid (s, s') \in E\}$ the set of successors of s by edges in E. We assume that graphs are deadlock-free, i.e., for all $s \in S$, $\mathsf{Succ}(s) \neq \emptyset$. We denote by W the largest absolute weight that appears in the graph. We assume that weights are encoded in binary and denote by $V = \lceil \log_2 W \rceil$ the number of bits of their encoding. Throughout this thesis,

we differentiate between pseudo-polynomial algorithms (polynomial in W) and truly-polynomial algorithms (polynomial in V).

Definition 2.2 (Play). A *play* in $\mathcal{G}$ from initial state $s_{\mathsf{init}} \in S$ is an infinite sequence of states $\pi = s_0 s_1 s_2 \dots$ such that $s_0 = s_{\mathsf{init}}$ and $(s_i, s_{i+1}) \in E$ for all $i \geq 0$.

The *prefix* up to the n-th state of π is the finite sequence $\pi(n) = s_0 s_1 \dots s_n$. We resp. denote the first and last states of the prefix by $\mathsf{First}(\pi(n)) = s_0$ and $\mathsf{Last}(\pi(n)) = s_n$. For a play π, we naturally extend the notation to $\mathsf{First}(\pi)$. The set of plays of $\mathcal{G}$ is denoted by $\mathsf{Plays}(\mathcal{G})$ and the corresponding set of prefixes is denoted by $\mathsf{Prefs}(\mathcal{G})$. Given a play $\pi \in \mathsf{Plays}(\mathcal{G})$, we denote by $\mathsf{Inf}(\pi) \subseteq S$ the set of states that are visited infinitely often along the play. The infinite suffix of a play starting in s_n is denoted $\pi(n, \infty)$.

Definition 2.3 (Value of play). Given a function $f \colon \mathsf{Plays}(\mathcal{G}) \to \mathbb{R} \cup \{-\infty, \infty\}$, the *value* of a play π is denoted by $f(\pi)$.

For the moment, we focus on graphs with one-dimension weights and one-dimension value functions. In Part II, we lift this restriction and consider vectors of weights and multi-dimension value functions.

Remark 2.4. We use the term *value* to characterize payoffs associated to plays, by analogy with the *expected value* in a stochastic setting, which we discuss in the following. In particular, this is not to be confused with the *minimax value* of a game [vN28, OR94]. This classical game-theoretic concept represents the rational outcome of a game when both players play optimally. In this work, it corresponds to the *optimal worst-case value*. ◁

2.1.2 Two-Player Games

In this thesis, we focus on *two-player turn-based games*. We denote the two *players* by $\mathcal{P}_1$ and $\mathcal{P}_2$.

Definition 2.5 (Game). A finite *game* is a tuple $G = (\mathcal{G}, S_1, S_2)$ composed of (i) a finite weighted graph $\mathcal{G} = (S, E, w)$; and (ii) a partition of its states S into S_1 and S_2 that resp. denote the sets of states belonging to $\mathcal{P}_1$ and $\mathcal{P}_2$.

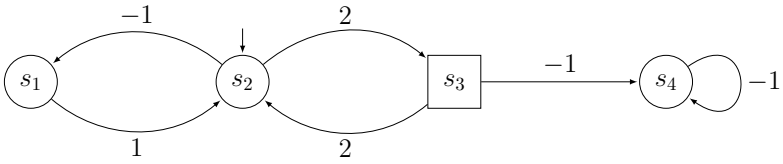


Figure 2.1: Simple two-player game over a weighted graph.

When explicit reference to the underlying graph is not necessary, we simply denote the game by $G = (S_1, S_2, E, w)$. A game is *one-player* if $S_2 = \emptyset$.

Example 2.6. A simple two-player game is presented in Fig. 2.1. States of $\mathcal{P}_1$ are depicted by circle vertices, states of $\mathcal{P}_2$ by square vertices. The initial state is indicated by the small arrow pointing to s_2. Edges are labeled with integer weights. ◁

A prefix $\pi(n)$ of a play π belongs to $\mathcal{P}_i$, $i \in \{1, 2\}$, if $\mathsf{Last}(\pi(n)) \in S_i$. The set of prefixes that belong to $\mathcal{P}_i$ is denoted by $\mathsf{Prefs}_i(G)$. We denote by $|G|$ the size of a game, defined as a polynomial function of $|S|$, $|E|$ and $V = \lceil \log_2 W \rceil$.

Remark 2.7. Recent research effort has been put into extending the two-player turn-based setting in two directions. First, multi-player non-zero-sum games have been studied in order to model complex systems composed of multiple components, with objectives that are not necessarily antagonistic (see for example De Pril's thesis [De 13]). An important part of the effort resides in defining meaningful solution concepts. For example, the seminal notion of Nash equilibrium [Nas50] has been refined in several ways (subgame perfect equilibria, secure equilibria, etc) [CHJ04, GU08, BMR14]. Second, games where players choose actions concurrently have been studied for a variety of winning objectives. See for example [dAHK98, Cha07a, BBMU12, KLST12].

Throughout this thesis, we focus on extending the power and expressiveness of the quantitative framework for the two-player turn-based case. Adapting our results to concurrent and/or multi-player games is an interesting long-term perspective. ◁

2.1.3 Strategies

Probability distributions. Given a finite set A, a (rational) *probability distribution* on A is a function $d\colon A \to [0, 1] \cap \mathbb{Q}$ such that $\sum_{a \in A} d(a) = 1$. We

denote the set of probability distributions on A by $\mathcal{D}(A)$. The *support* of the probability distribution d on A is $\mathsf{Supp}(d) = \{a \in A \mid d(a) > 0\}$.

Strategies. When the pebble is on a state belonging to $\mathcal{P}_i$, this player chooses how to move it on the graph according to his strategy.

Definition 2.8 (Strategy). Let $G = (S_1, S_2, E, w)$ be a two-player game, a *strategy* for $\mathcal{P}_i$, $i \in \{1, 2\}$, is a function $\lambda_i \colon \mathsf{Prefs}_i(G) \to \mathcal{D}(S)$ such that for all $\rho \in \mathsf{Prefs}_i(G)$, we have $\mathsf{Supp}(\lambda_i(\rho)) \subseteq \mathsf{Succ}(\mathsf{Last}(\rho))$.

A strategy is called *pure* if it is deterministic, i.e., if its support is a singleton for all prefixes. Otherwise it is said to be *randomized*. When a strategy λ_i of $\mathcal{P}_i$ is pure, we sometimes simplify its notation and write $\lambda_i(\rho) = s$ instead of $\lambda_i(\rho)(s) = 1$, for any $\rho \in \mathsf{Prefs}_i(G)$ and the unique state $s \in \mathsf{Supp}(\lambda_i(\rho))$.

A strategy λ_i for $\mathcal{P}_i$ has *finite memory* if it can be encoded by a finite state machine with stochastic outputs, called *stochastic output Moore machine*.

Definition 2.9 (Stochastic output Moore machine). A *stochastic output Moore machine* (SOMM) is a tuple $\mathcal{M}(\lambda_i) = (\mathsf{Mem}, \mathsf{m}_0, \alpha_\mathsf{u}, \alpha_\mathsf{n})$, where (i) Mem is a finite set of memory elements, (ii) $\mathsf{m}_0 \in \mathsf{Mem}$ is the initial memory element, (iii) $\alpha_\mathsf{u} \colon \mathsf{Mem} \times S \to \mathsf{Mem}$ is the update function, and (iv) $\alpha_\mathsf{n} \colon \mathsf{Mem} \times S_i \to \mathcal{D}(S)$ is the next-action function.

If the game is in $s \in S_i$ and $\mathsf{m} \in \mathsf{Mem}$ is the current memory element, then the strategy chooses s', the next state of the game, according to the probability distribution $\alpha_\mathsf{n}(\mathsf{m}, s)$. When the game leaves a state $s \in S$, the memory is updated to $\alpha_\mathsf{u}(\mathsf{m}, s)$. Formally, $(\mathsf{Mem}, \mathsf{m}_0, \alpha_\mathsf{u}, \alpha_\mathsf{n})$ defines the strategy λ_i such that $\lambda_i(\rho \cdot s) = \alpha_\mathsf{n}(\hat{\alpha}_\mathsf{u}(\mathsf{m}_0, \rho), s)$ for all $\rho \in \mathsf{Prefs}(G)$ and $s \in S_i$, where $\hat{\alpha}_\mathsf{u}$ extends α_u to sequences of states starting from m_0 as expected.

Note that pure finite-memory strategies have deterministic next-action functions. A strategy is *memoryless* if $|\mathsf{Mem}| = 1$, i.e., it does not depend on the history but only on the current state of the game.

Example 2.10. To illustrate those notions, consider the game in Fig. 2.2. It is one-player and unweighted. Observe that s_b is the only state where $\mathcal{P}_1$ has an actual choice. We intuitively define strategy λ_1 for $\mathcal{P}_1$ as follows. If the last visited state was s_a, then choose s_a with probability $1/3$ and state s_c with probability $2/3$. If the last visited state was s_c, then choose s_a or s_c with equal

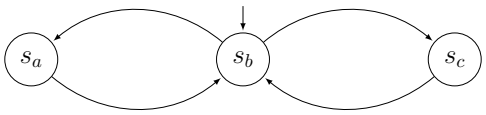


Figure 2.2: Simple one-player game over an unweighted graph.

probability. The first time you play, choose as if s_a was visited. This strategy is both finite-memory and randomized.

We encode λ_1 as a SOMM $\mathcal{M}(\lambda_1) = (\mathsf{Mem}, \mathsf{m}_0, \alpha_\mathsf{u}, \alpha_\mathsf{n})$. The two memory elements of $\mathsf{Mem} = \{\mathsf{m}_a, \mathsf{m}_c\}$ respectively represent that the last visited state between $\{s_a, s_c\}$ was s_a or s_c. The initial memory element is $\mathsf{m}_0 = \mathsf{m}_a$. The update and next-action functions are defined as follows:

$$\alpha_\mathsf{u} := \begin{cases} (\mathsf{m}_a, s_a) \mapsto \mathsf{m}_a \\ (\mathsf{m}_a, s_b) \mapsto \mathsf{m}_a \\ (\mathsf{m}_a, s_c) \mapsto \mathsf{m}_c \\ (\mathsf{m}_c, s_a) \mapsto \mathsf{m}_a \\ (\mathsf{m}_c, s_b) \mapsto \mathsf{m}_c \\ (\mathsf{m}_c, s_c) \mapsto \mathsf{m}_c \end{cases}, \qquad \alpha_\mathsf{n} := \begin{cases} (\mathsf{m}_a, s_a) \mapsto s_b \\ (\mathsf{m}_a, s_b) \mapsto \{(s_a, 1/3), (s_c, 2/3)\} \\ (\mathsf{m}_a, s_c) \mapsto s_b \\ (\mathsf{m}_c, s_a) \mapsto s_b \\ (\mathsf{m}_c, s_b) \mapsto \{(s_a, 1/2), (s_c, 1/2)\} \\ (\mathsf{m}_c, s_c) \mapsto s_b \end{cases}.$$

This SOMM is represented in Fig. 2.3. Edges of the machine are labeled following the format *input*/*output*. ◁

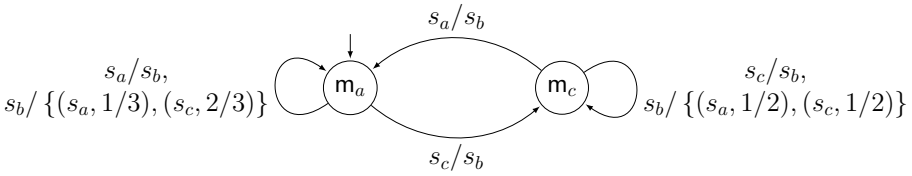


Figure 2.3: Stochastic output Moore machine for the strategy of Ex. 2.10.

We denote by $\Lambda_i(G), \Lambda_i^F(G), \Lambda_i^{PF}(G), \Lambda_i^M(G), \Lambda_i^{PM}(G)$ and $\Lambda_i^{RM}(G)$ the sets of general (i.e., possibly randomized and infinite-memory), finite-memory, pure finite-memory, memoryless, pure memoryless and randomized memoryless strategies for player $\mathcal{P}_i$ on the game G. We do not write G in this notation when the context is clear. A play π is said to be *consistent* with a strategy $\lambda_i \in \Lambda_i$ if for

all $n \geq 0$ such that $\mathsf{Last}(\pi(n)) \in S_i$, we have $\mathsf{Last}(\pi(n+1)) \in \mathsf{Supp}(\lambda_i(\pi(n)))$.

Remark 2.11. We use the classical model of strategies with deterministic memory updates and a possibly stochastic next-action function. This definition is widely used in the literature on games for verification and synthesis.

Some recent publications have explored the use of stochastic memory updates, with interesting results [BBC+11, CFK+12, BBC+13, CFK+13a]. From an implementation viewpoint, it is however necessary to consider representations of the involved probability distributions when comparing the memory usage of strategies. Note that this is also true for the next-action functions in both models of strategies.

On a related note, the reader familiar with traditional game theory may notice that our strategies, as described in Def. 2.8, correspond to *behavioral strategies*. That is, we put the randomization on the choice of actions. This is in contrast to *mixed strategies*, which put the randomization over pure strategies. Intuitively, behavioral strategies throw a dice whenever an action must be chosen whereas mixed strategies throw a dice once before starting the game to choose a pure strategy. A third notion, *general strategies*, combines both possibilities. In a setting of perfect recall (perfect information about the past, which is the case in this thesis), the well-known Kuhn's theorem states that all three variants have the same power [Aum64]. In other words, if a player can win with any of those strategy types, he can also win when restricted to any of the other two. Nevertheless, the choice of the model may still impact the complexity of the strategies needed to win. In particular, Cristau, David and Horn have shown some intruiguing examples where the memory needs vary greatly depending on this choice [CDH10b]. ◁

2.1.4 Markov Decision Processes

Definition 2.12 (Markov decision process). A finite *Markov decision process* (MDP) is a tuple $P = (\mathcal{G}, S_1, S_\Delta, \Delta)$ where (i) $\mathcal{G} = (S, E, w)$ is a finite weighted graph, (ii) S_1 and S_Δ define a partition of the set of states S into states of $\mathcal{P}_1$ and *stochastic states*, and (iii) $\Delta \colon S_\Delta \to \mathcal{D}(S)$ is the transition function that, given a stochastic state $s \in S_\Delta$, defines the probability distribution $\Delta(s)$ over the possible successors of s, such that for all states $s \in S_\Delta$, $\mathsf{Supp}(\Delta(s)) \subseteq \mathsf{Succ}(s)$.

As for games, when explicit reference to the underlying graph is not necessary, we simply denote the MDP by $P = (S_1, S_\Delta, E, \Delta, w)$.

An MDP can be seen as a two-player game where $\mathcal{P}_1$ is playing against a stochastic adversary using a fixed randomized memoryless strategy Δ in states of the set S_Δ. Hence MDPs are sometimes referred to as $1\frac{1}{2}$-player games in the literature. Notions of prefixes and strategies are naturally extended to MDPs.

2.1.5 Markov Chains

Definition 2.13 (Markov chain). A finite *Markov chain* (MC) is a tuple $M = (\mathcal{G}, \delta)$ where (i) $\mathcal{G} = (S, E, w)$ is a finite weighted graph; and (ii) $\delta \colon S \to \mathcal{D}(S)$ is the transition function that, given a state $s \in S$, defines the probability distribution $\delta(s)$ over the possible successors of s, such that for all states $s \in S$, $\mathsf{Supp}(\delta(s)) \subseteq \mathsf{Succ}(s)$.

As usual, when explicit reference to the underlying graph is not necessary, we simply denote the MC by $M = (S, E, \delta, w)$.

An MC can be seen as an MDP where the strategy of $\mathcal{P}_1$ is also fixed, or as a game where both strategies are fixed.

Probability measure and expected value. In a Markov chain $M = (\mathcal{G}, \delta)$, an *event* refers to a measurable set of plays $\mathcal{A} \subseteq \mathsf{Plays}(\mathcal{G})$. Every event has a uniquely defined probability [Var85] (Carathéodory's extension theorem induces a unique probability measure on the Borel σ-algebra over $\mathsf{Plays}(\mathcal{G})$). We denote by $\mathbb{P}^M_{s_{\mathsf{init}}}(\mathcal{A})$ the *probability* that a play belongs to $\mathcal{A}$ when the Markov chain M starts in $s_{\mathsf{init}} \in S$ and is executed for an infinite number of steps.

Given a measurable value function $f \colon \mathsf{Plays}(\mathcal{G}) \to \mathbb{R} \cup \{-\infty, \infty\}$, we denote by $\mathbb{E}^M_{s_{\mathsf{init}}}(f)$ the *expected value* or *expectation* of f over a play starting in s_{init}. The σ-algebra is defined through cylinder sets of prefixes: each prefix ρ defines a set of plays π such that ρ is a prefix of π [BK08]. Hence, the notions of probability and expected value can naturally be used over prefixes by considering the plays belonging to their cylinder set.

Example 2.14. Consider the Markov chain M depicted in Fig. 2.4. We use diamond vertices to represent stochastic states, and annotate outgoing edges with probabilities according to the transition function. Let event $\mathcal{A}$ be the set of plays that visit s_1 at least three times. It is easy to see that $\mathbb{P}^M_{s_1}(\mathcal{A}) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$

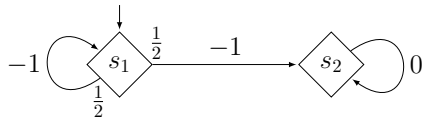


Figure 2.4: Simple Markov chain. Unlabeled edges have probability one.

(the first visit is granted by the initial edge). Now consider event $\mathcal{A}'$, defined as the set of plays that visit s_1 infinitely often. This event has probability zero: $\mathbb{P}^M_{s_1}(\mathcal{A}') = \lim_{n\to\infty} \left(\frac{1}{2}\right)^n = 0$. ◁

2.1.6 Outcomes

Projections. Given a set A_i, $1 \leq i \leq k$ of a cartesian product $A_1 \times \ldots \times A_k$, we define the *projection* over A_i, denoted by $\mathsf{proj}_{A_i} \colon A_1 \times \ldots \times A_k \to A_i$, as the mapping from elements $\overline{a} = (a_1, \ldots, a_k)$ to $\mathsf{proj}_{A_i}(\overline{a}) = a_i$.

Outcomes in MCs. Let $M = (\mathcal{G}, \delta)$ be a Markov chain, with $\mathcal{G} = (S, E, w)$ its underlying graph. Given an initial state $s_{\mathsf{init}} \in S$, we define the set of its possible *outcomes* as

$$\mathsf{Outs}_M(s_{\mathsf{init}}) = \big\{\pi = s_0 s_1 s_2 \ldots \in \mathsf{Plays}(\mathcal{G}) \mid s_0 = s_{\mathsf{init}} \wedge \forall\, n \in \mathbb{N},\ s_{n+1} \in \mathsf{Supp}(\delta(s_n))\big\}.$$

Note that if δ is deterministic (i.e., if the support is a singleton) in all states, we obtain a unique play $\pi = s_0 s_1 s_2 \ldots$ as the unique possible outcome.

Outcomes in games. Let $G = (\mathcal{G}, S_1, S_2)$ be a two-player game, with $\mathcal{G} = (S, E, w)$ its underlying graph. Given two strategies, $\lambda_1 \in \Lambda_1$ and $\lambda_2 \in \Lambda_2$, and an initial state $s_{\mathsf{init}} \in S$, we extend the notion of outcomes as follows:

$$\mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2) = \big\{\pi = s_0 s_1 s_2 \ldots \in \mathsf{Plays}(\mathcal{G}) \mid s_0 = s_{\mathsf{init}} \wedge \pi \text{ is consistent with } \lambda_1 \text{ and } \lambda_2\big\}.$$

Observe that when fixing the strategies, we obtain an MC denoted by $G[\lambda_1, \lambda_2]$. This MC is finite if both λ_1 and λ_2 are finite-memory strategies. Let $\mathcal{M}(\lambda_1) = (\mathsf{Mem}_1, \mathsf{m}_1, \alpha^1_{\mathsf{u}}, \alpha^1_{\mathsf{n}})$ and $\mathcal{M}(\lambda_2) = (\mathsf{Mem}_2, \mathsf{m}_2, \alpha^2_{\mathsf{u}}, \alpha^2_{\mathsf{n}})$ be the SOMMs of two such strategies. The set of states of the resulting MC is obtained through the product

of the memory elements of the strategies given as SOMMs and the states of the game, i.e., $S \times \mathsf{Mem}_1 \times \mathsf{Mem}_2$. Its transition function is defined based on the distributions prescribed by the strategies and in order to accurately account for the memory updates.

Notice that the outcomes of G and $G[\lambda_1, \lambda_2]$ are different objects by nature: the former are plays on a graph defined by the set of states S while the latter are plays on a graph defined by $S \times \mathsf{Mem}_1 \times \mathsf{Mem}_2$. Still, there exists a bijection between outcomes of the MC and their *traces* in the initial game, thanks to the projection operator (Lemma 2.16).

Example 2.15. Let G be the game depicted in Fig. 2.2. This game is actually an MDP (with no stochastic state) since the set of states of $\mathcal{P}_2$ is empty. Let λ_1 be the randomized finite-memory strategy presented in Ex. 2.10. The corresponding SOMM $\mathcal{M}(\lambda_1)$ is illustrated in Fig. 2.3.

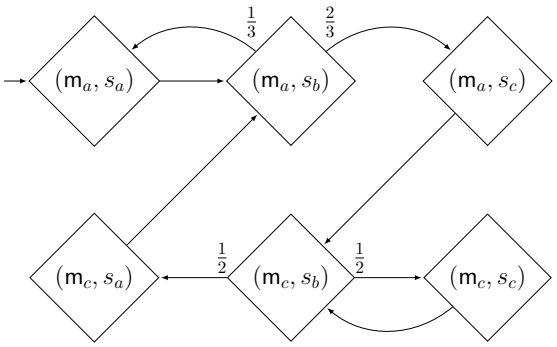


Figure 2.5: Markov chain obtained by product of the one-player game of Fig. 2.2 and the SOMM of Fig. 2.3.

Applying strategy λ_1 on G induces a Markov chain, obtained via the product of G and $\mathcal{M}(\lambda_1)$. This MC is depicted in Fig. 2.5. While the graph is now based on the state space $\mathsf{Mem} \times S$, it is easy to reason on traces in the original graph by projecting plays over the set S. ◁

Lemma 2.16 ([BFRR13, Lemma 1]). *Let $G = (\mathcal{G}, S_1, S_2)$ be a game, with $\mathcal{G} = (S, E, w)$ its underlying graph. Let $\lambda_1 \in \Lambda_1^F$ and $\lambda_2 \in \Lambda_2^F$ be the finite-memory strategies of the players. Then there is a bijection between outcomes in G and outcomes in the resulting Markov chain $G[\lambda_1, \lambda_2]$.*

Proof. Let $s_{\mathsf{init}} \in S$ be the initial state of the game, $\mathcal{M}(\lambda_1) = (\mathsf{Mem}_1, \mathsf{m}_1, \alpha^1_{\mathsf{u}}, \alpha^1_{\mathsf{n}})$ and $\mathcal{M}(\lambda_2) = (\mathsf{Mem}_2, \mathsf{m}_2, \alpha^2_{\mathsf{u}}, \alpha^2_{\mathsf{n}})$ be the SOMMs.

An outcome in $G[\lambda_1, \lambda_2]$ is a sequence of states from $S \times \mathsf{Mem}_1 \times \mathsf{Mem}_2$. Its projection on the set S is unique and defines the outcome in the sense of G.

Conversely, consider an outcome in G: it is of the form $s_0 s_1 s_2 \ldots \in S^\omega$, with $s_0 = s_{\mathsf{init}}$. We claim there is a unique corresponding outcome in $G[\lambda_1, \lambda_2]$, written $(s_0, \mathsf{m}^0_1, \mathsf{m}^0_2)(s_1, \mathsf{m}^1_1, \mathsf{m}^1_2)\ldots \in S \times \mathsf{Mem}_1 \times \mathsf{Mem}_2$, with $(s_0, \mathsf{m}^0_1, \mathsf{m}^0_2) = (s_{\mathsf{init}}, \mathsf{m}_1, \mathsf{m}_2)$. Indeed, it suffices to see that the update functions of the SOMMs, α^1_{u} and α^2_{u}, are deterministic functions. Hence, it is easy to reconstruct the outcome of $G[\lambda_1, \lambda_2]$ based on its projection on S as it suffices to apply the effect of the update functions on the memory at each step. □

Hence, we obtain the following equality:

$$\mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2) = \mathsf{proj}_S\left(\mathsf{Outs}_{G[\lambda_1,\lambda_2]}((s_{\mathsf{init}}, \mathsf{m}_1, \mathsf{m}_2))\right).$$

Based on this, and for the sake of readability, we abuse the notation and write $\mathsf{Outs}_{G[\lambda_1,\lambda_2]}(s_{\mathsf{init}})$ equivalently to refer to this set of outcomes. Similar abuse is taken for value functions and initial states.

Back to the outcomes of the game: note that if both strategies λ_1 and λ_2 are pure, the resulting Markov chain only involves Dirac distributions (δ is deterministic) and the set $\mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2)$ is composed of a unique play $\pi = s_0 s_1 s_2 \ldots$ such that for all $n \geq 0$, $i \in \{1, 2\}$, if $s_n \in S_i$, then we have $\lambda_i(s_n) = s_{n+1}$.

Outcomes in MDPs. Let $P = (\mathcal{G}, S_1, S_\Delta, \Delta)$ be a Markov decision process, with $\mathcal{G} = (S, E, w)$ its underlying graph. Again, we can fix the strategy λ_1 of $\mathcal{P}_1$ and obtain the Markov chain $P[\lambda_1]$. Let $\mathcal{M}(\lambda_1) = (\mathsf{Mem}, \mathsf{m}_0, \alpha_{\mathsf{u}}, \alpha_{\mathsf{n}})$, assuming the strategy is finite-memory. The set of outcomes starting in $s_{\mathsf{init}} \in S$ is defined as $\mathsf{Outs}_P(s_{\mathsf{init}}, \lambda_1) = \mathsf{proj}_S\left(\mathsf{Outs}_{P[\lambda_1]}((s_{\mathsf{init}}, \mathsf{m}_0))\right)$. Again, we abuse the notation and write $\mathsf{Outs}_{P[\lambda_1]}(s_{\mathsf{init}})$ equivalently.

Finally, back to the two-player game G, if we fix the strategy λ_i of only one player $\mathcal{P}_i$, $i \in \{1, 2\}$, we obtain not a Markov chain, but a Markov decision process for the remaining player $\mathcal{P}_{3-i}$. This MDP is denoted by $G[\lambda_i]$. We define

its set of outcomes as

$$\mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_i) = \bigcup_{\lambda_{3-i} \in \Lambda_{3-i}} \mathsf{Outs}_{G[\lambda_i]}(s_{\mathsf{init}}, \lambda_{3-i}) = \bigcup_{\lambda_{3-i} \in \Lambda_{3-i}} \mathsf{Outs}_{G[\lambda_1,\lambda_2]}(s_{\mathsf{init}}).$$

2.1.7 Attractors and Submodels

Attractors. Given a game $G = (S_1, S_2, E, w)$, the *attractor* for $\mathcal{P}_1$ of a set $A \subseteq S$ in G is denoted by $\mathsf{Attr}_G^{\mathcal{P}_1}(A)$ and computed as the fixed point of the sequence

$$\begin{aligned}\mathsf{Attr}_G^{\mathcal{P}_1,\,n+1}(A) = \mathsf{Attr}_G^{\mathcal{P}_1,\,n}(A) \\ \cup \{s \in S_1 \mid \exists\,(s,t) \in E,\, t \in \mathsf{Attr}_G^{\mathcal{P}_1,\,n}(A)\} \\ \cup \{s \in S_2 \mid \forall\,(s,t) \in E,\, t \in \mathsf{Attr}_G^{\mathcal{P}_1,\,n}(A)\},\end{aligned}$$

with $\mathsf{Attr}_G^{\mathcal{P}_1,\,0}(A) = A$. The attractor $\mathsf{Attr}_G^{\mathcal{P}_1}(A)$ is exactly the set of states from which $\mathcal{P}_1$ can ensure to reach A no matter what $\mathcal{P}_2$ does. That is,

$$\begin{aligned}\mathsf{Attr}_G^{\mathcal{P}_1}(A) = \big\{s \in S \mid \exists\,\lambda_1 \in \Lambda_1(G),\, \forall\,\lambda_2 \in \Lambda_2(G), \\ \forall\,\pi = s_0 s_1 s_2 \ldots \in \mathsf{Outs}_G(s, \lambda_1, \lambda_2),\, s_0 = s,\, \exists\, i \in \mathbb{N},\, s_i \in A\big\}\end{aligned}$$

The attractor $\mathsf{Attr}_G^{\mathcal{P}_2}(A)$ for $\mathcal{P}_2$ is defined symmetrically.

Subgraphs, subgames and sub-MDPs. Given a graph $\mathcal{G} = (S, E, w)$ and a subset of states $A \subseteq S$, we define the induced subgraph naturally. We denote it by $\mathcal{G} \restriction A = (A, E \cap (A \times A), w)$. Subgames and sub-MDPs are defined similarly by considering their induced subgraphs. It is to note that subgames and sub-MDPs can only be properly defined if the induced subgraphs contain no deadlock and if the transition functions remain well-defined in the case of MDPs (i.e., if the probabilities on outgoing edges still sum up to one in all stochastic states of the sub-MDP).

2.2 Objectives and Decision Problems

We present the following notions with regard to games w.l.o.g. as MDPs can be seen as games where $\mathcal{P}_2$ has fixed his strategy.

Winning objectives. Let G be a game. A *winning objective* for $\mathcal{P}_i$ in G is a predefined set of plays $\phi_i \subseteq \mathsf{Plays}(G)$. A play $\pi \in \mathsf{Plays}(G)$ is considered winning for $\mathcal{P}_i$ if $\pi \in \phi_i$. We present several classical objectives in Section 2.3.

Our work is focused on *zero-sum* games. That is, $\phi_1 \cup \phi_2 = \mathsf{Plays}(G)$ and $\phi_1 \cap \phi_2 = \emptyset$. In practice, a play is winning for $\mathcal{P}_1$ if and only if it is losing for $\mathcal{P}_2$. In the following, we thus take the point of view of $\mathcal{P}_1$, and state the objectives accordingly. We are interested in strategies of $\mathcal{P}_1$ that induce winning plays.

2.2.1 Qualitative Objectives

Qualitative objectives express Boolean properties on plays such as reachability, liveness, etc. A play either satisfies the property (it is winning) or it does not (it is losing). We consider two semantics: the *surely winning* and the *almost-surely winning* semantics.

Definition 2.17 (Surely winning). Given an initial state $s_{\mathsf{init}} \in S$, a strategy $\lambda_1 \in \Lambda_1$ for $\mathcal{P}_1$ is *surely winning* for an objective ϕ in G if for all plays $\pi \in \mathsf{Plays}(G)$ that are consistent with λ_1 from s_{init}, we have $\pi \in \phi$.

In other words, a surely winning strategy ensures winning plays against any strategy of $\mathcal{P}_2$. That is, λ_1 is surely winning from s_{init} if $\mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1) \subseteq \phi$.

When in an MDP, or when at least one of the players plays a randomized strategy, the notion of sure winning may be too restrictive and inadequate, as the set of consistent plays that do not belong to ϕ may have zero probability measure. Therefore, it is useful to reason about the *satisfaction probability*.

Definition 2.18 (Almost-surely winning). Let $\lambda_1 \in \Lambda_1$ and $s_{\mathsf{init}} \in S$. Given a threshold $\alpha \in [0, 1]$ and a measurable objective $\phi \subseteq \mathsf{Plays}(G)$, *$\alpha$-satisfaction* asks that for all $\lambda_2 \in \Lambda_2$, we have $\mathbb{P}_{s_{\mathsf{init}}}^{G[\lambda_1,\lambda_2]}(\phi) \geq \alpha$. If λ_1 satisfies ϕ with probability $\alpha = 1$, we say that λ_1 is *almost-surely winning* for ϕ in G.

Both for surely and almost-surely winning semantics, the associated *decision problems* ask to decide, given a game, an objective and an initial state, if $\mathcal{P}_1$ has a corresponding winning strategy.

2.2.2 Quantitative Objectives

Given a value function f and a rational threshold $v \in \mathbb{Q}$, we naturally define a corresponding winning objective $\phi_{f,\sim}(v) = \{\pi \in \mathsf{Plays}(G) \mid f(\pi) \sim v\}$, for any $\sim$ in $\{>, \geq, =, <, \leq\}$. Note that we restrict to rational thresholds due to computational problems arising with irrational ones.

As for qualitative objectives, surely winning and almost-surely winning semantics can be used for quantitative objectives. A third semantics is also of particular interest in the case of MDPs:[1] the *expected value* or *expectation* semantics.

In the quantitative context, it is common to present the surely winning and expected value semantics through their associated decision problems, namely the *worst-case threshold problem* and the *expected value threshold problem*.

Definition 2.19 (Worst-case threshold problem). Given a two-player game $G = (S_1, S_2, E, w)$, an initial state $s_{\mathsf{init}} \in S$, a value function $f \colon \mathsf{Plays}(G) \to \mathbb{R} \cup \{-\infty, \infty\}$, and a rational threshold $\mu \in \mathbb{Q}$, the *worst-case threshold problem* asks to decide if $\mathcal{P}_1$ has a strategy $\lambda_1 \in \Lambda_1$ such that

$$\forall\, \lambda_2 \in \Lambda_2, \forall\, \pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2),\ f(\pi) \geq \mu.$$

Definition 2.20 (Expected value threshold problem). Given an MDP $P = (S_1, S_\Delta, E, \Delta, w)$, an initial state $s_{\mathsf{init}} \in S$, a measurable value function $f \colon \mathsf{Plays}(P) \to \mathbb{R} \cup \{-\infty, \infty\}$, and a rational threshold $\nu \in \mathbb{Q}$, the *expected value threshold problem* asks to decide if $\mathcal{P}_1$ has a strategy $\lambda_1 \in \Lambda_1$ such that

$$\mathbb{E}^{P[\lambda_1]}_{s_{\mathsf{init}}}(f) \geq \nu.$$

Remark 2.21. It is to note that there exist other interesting decision problems, and some specific to particular value functions. Our presentation here is not exhaustive: it focuses on the most commonly studied problems, and those useful to understand our contributions. ◁

[1] Or games played over a stochastic arena. See for example [CMH08, CDH09, CFK+13b].

2.2.3 Synthesis Problem

Synthesis. In the previously defined decision problems, the question is to decide whether $\mathcal{P}_1$ has a winning strategy (with respect to the appropriate winning semantics). For all the studied models, we are also interested in constructing such winning strategies if they exist. This is known as the *synthesis problem*.

Our ultimate goal is to provide efficiently implementable controllers. Hence, we favor strategies that are considered simple to implement and/or using a minimal amount of resources. Formally, we notably study the memory requirements of winning strategies, both for $\mathcal{P}_1$ and $\mathcal{P}_2$. For example, we will see that for most classical objectives, *memoryless strategies* are sufficient. That is, if $\mathcal{P}_i$ has a winning strategy, then he has one which is memoryless.

Finite memory. When considering more complex objectives (see Parts II, III and IV), strategies with infinite memory may be required in general. In that case, it is of interest to restrict $\mathcal{P}_1$ to *finite-memory strategies*, as they correspond to strategies that can be implemented in practice.

2.2.4 Determinacy and Optimality

Determinacy. Given an initial state $s_{\mathsf{init}} \in S$, we are looking for a winning strategy for $\mathcal{P}_1$, or a winning strategy for $\mathcal{P}_2$ (i.e., a strategy that ensures a losing play for $\mathcal{P}_1$). In full generality, there may exist states where none of the players have a winning strategy: neither can $\mathcal{P}_1$ ensure a winning play nor can $\mathcal{P}_2$ ensure a losing one. In other words, $\mathcal{P}_2$ may have a counter-strategy for each strategy of $\mathcal{P}_1$, but no unique strategy that counters all strategies of $\mathcal{P}_1$.

Given some objective for $\mathcal{P}_i$, we say that a state is a *winning state* for $\mathcal{P}_i$ if $\mathcal{P}_i$ has a winning strategy from this state. Games where all states are either winning for $\mathcal{P}_1$ or winning for $\mathcal{P}_2$ are called *determined*. Determinacy is an important concept: most interesting questions only apply on games that are determined.

Classes of determined games are well-studied. Fortunately, very general results by Martin guarantee determinacy for our game models with Borel objectives [Mar75, Mar98]. The objectives studied in this thesis are Borel, hence Martin's theorems apply and guarantee determinacy.

Optimal and ε-optimal strategies. Consider a game G, a value function $f\colon \mathsf{Plays}(G) \to \mathbb{R} \cup \{-\infty, \infty\}$ and a threshold problem (worst-case or expected

value). We assume w.l.o.g. that $\mathcal{P}_1$ aims at maximizing the value of plays. We classicaly define the *optimal value* of a threshold problem as the $\sup_{\lambda_1 \in \Lambda_1} \inf_{\lambda_2 \in \Lambda_2}$ over thresholds that can be ensured.

For example, given a game G with a one-dimension quantitative objective $\phi_{f,\geq}(v)$ and an initial state $s_{\mathsf{init}} \in S$, we define its *worst-case optimal value* as

$$\underline{\mathsf{val}}(\phi_{f,\geq}) = \sup_{\lambda_1 \in \Lambda_1} \inf_{\lambda_2 \in \Lambda_2} \left\{ v \mid \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2) \subseteq \phi_{f,\geq}(v) \right\}.$$

A strategy for $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) is then said to be *optimal* for this problem if it ensures a threshold higher (resp. lower) than or equal to the threshold ensured by any other strategy of the same player. That is, if it ensures the optimal value.

Remark 2.22. The above definition of value is the *lower value* (value for $\mathcal{P}_1$). In full generality, the *upper value* (value for $\mathcal{P}_2$), defined by swapping the sup and inf, may be different. However, it is well-known that the lower and upper values are equal in games that are determined, as defined in Sect. 2.2.4. Since we only consider determined games, we do not distinguish them. ◁

For complex winning objectives, optimal strategies may not exist in general (the supremum need not be a maximum) or have prohibitive memory requirements (infinite memory). See for example [CHJ05, CD11, BMOU11] or the mean-payoff Büchi game that is depicted in Fig. 7.4. In that case, it is useful to look for *ε-optimal strategies*: strategies achieving a value $\underline{\mathsf{val}}(\phi_{f,\geq}) - \varepsilon$ for some given $\varepsilon > 0$. Observe that for all $\varepsilon > 0$, ε-optimal strategies (i.e., that achieve value $(\underline{\mathsf{val}}(\phi_{f,\geq}) - \varepsilon)$) always exist in all determined games. It is a consequence of the existence of a game optimal value, equal to both the upper and lower values, as sketched in Remark 2.22.

2.3 Some Classical Objectives

We present some classical objectives and important results that we use through this thesis. We are particularly interested in the complexity of decision problems and winning strategies. For quantitative objectives, we only discuss games with one-dimension weights, as we will extend this setting to multi-dimension weights in Part II.

2.3.1 Qualitative Objectives

We mention three important qualitative objectives: reachability, Büchi and parity. An overview of the corresponding results is presented in Table 2.1.

Reachability. Given a target set of states $T \subseteq S$, the *reachability objective* asks for plays that reach T, an event denoted by $\Diamond T$ according to the classical LTL formulation [Pnu77]. That is,

$$\mathsf{Reach}_G(T) = \{\pi = s_0 s_1 s_2 \ldots \in \mathsf{Plays}(G) \mid \exists\, i \in \mathbb{N},\, s_i \in T\}\,.$$

To win this objective, $\mathcal{P}_1$ must be able to force visiting a state of T at least once.

Example 2.23. Consider the game depicted in Fig. 2.1. Let $T = \{s_3\}$ be the target set and $s_{\mathsf{init}} = s_2$ be the initial state. A sure winning strategy for $\mathcal{P}_1$ is to take edge (s_2, s_3). Now assume the target set is $T' = \{s_4\}$. In that case, $\mathcal{P}_1$ has no sure winning strategy because $\mathcal{P}_2$ can always choose to take edge (s_3, s_2).

Now, consider the MC defined in Fig. 2.4. There is no choice for $\mathcal{P}_1$ but we can still check if the target set $T'' = \{s_2\}$ is reached eventually. Clearly, it is not reached surely: the play $(s_1)^\omega$ is a valid outcome and never reaches T''. Still, the unique play that never reaches the target set has a zero probability. Hence, the reachability objective is almost-surely satisfied. ◁

For games with the sure semantics, pure memoryless strategies suffice for both players. The set of winning states for $\mathcal{P}_1$ can be computed in linear time: it is simply the attractor $\mathsf{Attr}_G^{\mathcal{P}_1}(T)$. Deciding the winner in reachability games is P-complete [Bee80, Imm81]. For MDPs with the almost-sure semantics, pure memoryless strategies also suffice for the only player, $\mathcal{P}_1$. The decision problem can also be solved in polynomial time [CJH03, BK08]. For exemple, Chatterjee and (M.) Henzinger present efficient dynamic graph algorithms operating in $\mathcal{O}(|S|^{2/3} \cdot |E|)$-time in [CH11].

Büchi. Given a target set of states $T \subseteq S$, the *Büchi objective* asks for plays that visit T infinitely often, an event denoted by $\Box\Diamond T$ in LTL. That is,

$$\mathsf{Buchi}_G(T) = \{\pi = s_0 s_1 s_2 \ldots \in \mathsf{Plays}(G) \mid \mathsf{Inf}(\pi) \cap T \neq \emptyset\}\,.$$

To win this objective, $\mathcal{P}_1$ must be able to force repeated visits of the set T.

The Büchi objective generalizes the reachability one by making it a repeated

		reachability	Büchi	parity
GAMES *sure sem.*	complexity	P-complete		$\mathrm{UP} \cap \mathrm{coUP}$
	$\mathcal{P}_1$ mem.	pure memoryless		
	$\mathcal{P}_2$ mem.			
MDPS *almost-sure sem.*	complexity	P-complete		
	$\mathcal{P}_1$ mem.	pure memoryless		

Table 2.1: Overview of some known results for qualitative objectives in games (sure semantics) and MDPs (almost-sure semantics).

reachability objective. In particular, simple reachability can be encoded as a Büchi objective by transforming the states of the target set T into *absorbing states* (states with a single outgoing edge, which is a self-loop).

Example 2.24. Consider the game of Fig. 2.1. Let $T = \{s_3\}$ be the target set and $s_{\mathsf{init}} = s_2$ be the initial state. While $\mathcal{P}_1$ can ensure reaching T once with a simple attractor strategy, he cannot ensure repeated visits. Indeed, $\mathcal{P}_2$ can take edge (s_3, s_4) to prevent further visits of s_3. ◁

Both for games with sure semantics and MDPs with almost-sure semantics, pure memoryless strategies are sufficient. The classical algorithm for Büchi games has a time complexity of $\mathcal{O}(|S| \cdot |E|)$ [EJ91, McN93, Zie98]. Hence deciding the winner in Büchi games and MDPs is also P-complete. An algorithm running in $\mathcal{O}(|S|^2)$ has been recently introduced by Chatterjee and (M.) Henzinger [CH12].

Parity. Let G be a game extended with a *priority function* $p\colon S \to \mathbb{N}$. The *parity* of a play π is defined as $\mathsf{Par}(\pi) = \min\{p(s) \mid s \in \mathsf{Inf}(\pi)\}$. The *parity objective* requires that the minimum priority visited infinitely often be even.

$$\mathsf{Parity}_G = \{\pi = s_0 s_1 s_2 \ldots \in \mathsf{Plays}(G) \mid \mathsf{Par}(\pi) \bmod 2 = 0\}\,.$$

Parity is a canonical way to encode ω-regular objectives [Tho97]. In particular, when the set of priorities is restricted to $\{0, 1\}$, we have a Büchi objective.

Example 2.25. We consider the game depicted in Fig. 2.1 with a priority function $p\colon \{s_1 \mapsto 1, s_2 \mapsto 1, s_3 \mapsto 0, s_4 \mapsto 2\}$. Then, $\mathcal{P}_1$ can surely win by repeatedly

choosing edge (s_2, s_3). Indeed, either $\mathcal{P}_2$ never goes on the right, in which case the minimal priority seen infinitely often will be 0, or he eventually chooses the right edge. If he does so, the only state visited infinitely often is s_4, and its priority is even. Hence in both cases, consistent plays are winning for $\mathcal{P}_1$. $\lhd$

Again, pure memoryless strategies suffice in both games and MDPs, with the corresponding semantics. See [EJ88, Zie98] for games and [CY90, dA97, CJH04] for MDPs.

While the previous objectives all admit polynomial decision algorithms both in games and MDPs, the situation is different for parity. Almost-sure winning can still be decided in polynomial time for MDPs [CY90, CJH04]. For games, the problem is in NP $\cap$ coNP: pure memoryless optimal strategies exist for both players and one-player variants are solvable in polynomial time [KKV01]. Nonetheless, whether parity games are in P is a long-standing open problem. Parity games can be polynomially reduced to mean-payoff games, which also belong to NP $\cap$ coNP, as discussed in the following. This reduction is notably presented by Jurdziński in [Jur98], also showing that the complexity class can be refined to UP $\cap$ coUP (subclass of NP $\cap$ coNP restricted to unambiguous Turing machines [Pap94]). Note that parity games subsume modal μ-calculus model checking [EJS93].

2.3.2 Quantitative Objectives

We present several well-known value functions and associated decision problems: total-payoff, mean-payoff, shortest path and energy. An overview of the corresponding results is presented in Table 2.2.

Total-payoff. The *total-payoff* of a prefix $\rho = s_0 s_1 \dots s_n$ is defined by the value function $\mathsf{TP}(\rho) = \sum_{i=0}^{i=n-1} w((s_i, s_{i+1}))$. This is naturally extended to plays by considering the limit behavior. Since the total-payoff need not converge in general, we define two variants. The *infimum (resp. supremum) total-payoff* of a play π is $\underline{\mathsf{TP}}(\pi) = \liminf_{n\to\infty} \mathsf{TP}(\pi(n))$ (resp. $\overline{\mathsf{TP}}(\pi) = \limsup_{n\to\infty} \mathsf{TP}(\pi(n))$).

The sets of winning plays for the worst-case semantics for the infimum and

		TP	MP	SP	EG
GAMES *worst-case*	complexity	UP ∩ coUP		P	UP ∩ coUP
	$\mathcal{P}_1$ mem.	pure memoryless			
	$\mathcal{P}_2$ mem.				
MDPS *expected value*	complexity	P-complete			n/a
	$\mathcal{P}_1$ mem.	pure memoryless			

Table 2.2: Overview of some known results for quantitative objectives in games (worst-case semantics) and MDPs (expected value semantics). TP stands for total-payoff, MP for mean-payoff, SP for shortest path and EG for energy. In the energy context, the decision problem is the unknown initial credit problem.

supremum total-payoffs with threshold $\mu \in \mathbb{Q}$ are written as follows:

$$\mathsf{TotalInf}_G(\mu) = \{\pi \in \mathsf{Plays}(G) \mid \underline{\mathsf{TP}}(\pi) \geq \mu\},$$
$$\mathsf{TotalSup}_G(\mu) = \left\{\pi \in \mathsf{Plays}(G) \mid \overline{\mathsf{TP}}(\pi) \geq \mu\right\}.$$

The total-payoff objective models total accumulation (or consumption) of resources along infinite plays. It can be seen as a refinement of mean-payoff games for the particular case of optimal mean-payoff equal to zero.

Example 2.26. Consider Fig. 2.1. Assume $\mathcal{P}_1$ always chooses to go left. Then the only consistent outcome is $\pi = (s_2 s_1)^\omega$ and the sequence of partial sums is $-1, 0, -1, 0 \ldots$ We have that $\underline{\mathsf{TP}}(\pi) = -1$ and $\overline{\mathsf{TP}}(\pi) = 0$.

Let M be the MC described in Fig. 2.4. We are interested in computing the expected (infimum) total-payoff. Observe that with probability $1/2$ we reach the zero-loop in one step, with $1/4$ in two steps, and so on. Thus, we have that $\mathbb{E}^M_{s_{\mathsf{init}}}(\underline{\mathsf{TP}}) = -1 \cdot \sum_{n=0}^{\infty} \left(\frac{1}{2}\right)^n = -2$. ◁

One-dimension total-payoff value functions admit pure memoryless optimal strategies, both in games for both players (worst-case semantics) and in MDPs (expected value semantics) [FV97, GZ04]. In terms of complexity of the threshold problems, the situation is similar to the parity objective. The worst-case threshold problem for games is in UP ∩ coUP and no polynomial algorithm is known [GS09]. The expected value threshold problem can be solved in polynomial time through linear programming [FV97].

Mean-payoff. The *mean-payoff* of a prefix $\rho = s_0 s_1 \dots s_n$ is defined by the value function $\mathsf{MP}(\rho) = \frac{1}{n}\mathsf{TP}(\rho)$. Again, this is extended to plays by considering the limit behavior, and two variants coexist. For a play π, the *infimum mean-payoff* is $\underline{\mathsf{MP}}(\pi) = \liminf_{n\to\infty} \mathsf{MP}(\pi(n))$ and the *supremum mean-payoff* is $\overline{\mathsf{MP}}(\pi) = \limsup_{n\to\infty} \mathsf{MP}(\pi(n))$.

The sets of winning plays for the worst-case semantics for the infimum and supremum mean-payoffs with threshold $\mu \in \mathbb{Q}$ are written as follows:

$$\mathsf{MeanInf}_G(\mu) = \{\pi \in \mathsf{Plays}(G) \mid \underline{\mathsf{MP}}(\pi) \geq \mu\},$$
$$\mathsf{MeanSup}_G(\mu) = \left\{\pi \in \mathsf{Plays}(G) \mid \overline{\mathsf{MP}}(\pi) \geq \mu\right\}.$$

Observe that threshold μ can be taken equal to zero w.l.o.g. as we transform the weight function w to $b \cdot w - a$ for any threshold $\frac{a}{b}$, $a \in \mathbb{Z}$, $b \in \mathbb{N}_0 = \mathbb{N} \setminus \{0\}$.

Mean-payoff is a classical game-theory objective, modeling the long-run average reward or cost that a player can obtain over an infinite play. It is a powerful model: parity games (a fortiori modal μ-calculus model checking) can be reduced to mean-payoff games [Jur98].

The mean-payoff objective is *prefix-independent*: for all $\rho \in \mathsf{Prefs}(G)$, $\pi \in \mathsf{Plays}(G)$, we have that $\underline{\mathsf{MP}}(\rho{\cdot}\pi) = \underline{\mathsf{MP}}(\pi)$ (resp. $\overline{\mathsf{MP}}(\rho{\cdot}\pi) = \overline{\mathsf{MP}}(\pi)$). Intuitively, its value only depends on the long-run behavior and not on finite prefixes.

Example 2.27. Consider the game of Fig. 2.1. Recall that always taking left yields an infimum (resp. supremum) total-payoff of -1 (resp. 0). For the mean-payoff, the same outcome has a value of 0 for both variants of the limit.

Let M be the MC in Fig. 2.4. Consider the expected mean-payoff. As already discussed, the probability of eventually reaching state s_2 is one. Since the mean-payoff value of any play is finite (trivially bounded by $-W$ and W, in this specific case, by -1 and 0), plays that never reach s_2 do not impact the expectation. Moreover, as mean-payoff is prefix-independent, finite prefixes up to reaching s_2 are also negligible. We conclude that $\mathbb{E}^M_{s_1}(\underline{\mathsf{MP}}) = \mathbb{E}^M_{s_1}(\overline{\mathsf{MP}}) = 0$. ◁

For the specific case of one-dimension games (worst-case semantics), pure memoryless optimal strategies always exist for both players [LL69, EM79]. Deciding the winner is in UP ∩ coUP [KL93, ZP96, Jur98, GS09]. An interesting consequence of the memoryless determinacy is that both variants coincide in terms of decision problem: the limit exists for outcomes induced by pure finite-

memory strategies. This equivalence is no longer true when moving to multi-dimension games (see Chap. 4). Again, whether mean-payoff games are in P is a long-standing open problem which stays open despite many efforts [GKK88, Con93, ZP96, Pis99, LP07, BV07, BCD+11]. In MDPs with the expected value semantics, $\mathcal{P}_1$ can achieve the optimal expectation with a pure memoryless strategy [Put94,FV97,Gim07]. The corresponding threshold problem can be answered in polynomial time via linear programming.

Shortest path. This quantitative setting is a generalization of the classical shortest path problem over graphs (see for example [CGR96] for the graph setting). We assume game graphs where all weights are strictly positive (i.e., $w\colon E \to \mathbb{N}_0$): they represent costs that $\mathcal{P}_1$ wants to minimize. Given a target set of states $T \subseteq S$, we define the *truncated sum up to* T as $\mathsf{TS}_T\colon \mathsf{Plays}(\mathcal{G}) \to \mathbb{N} \cup \{\infty\}$, $\mathsf{TS}_T(\pi = s_0 s_1 s_2 \dots) = \sum_{i=0}^{n-1} w((s_i, s_{i+1}))$, with n the first index such that $s_n \in T$, and $\mathsf{TS}_T(\pi) = \infty$ if π never reaches any state in T. An example of a shortest path application is presented in Ex. 11.1.

The set of winning plays for the worst-case semantics for the shortest path objective with threshold $\mu \in \mathbb{N}$ and target set $T \subseteq S$ is written as follows:

$$\mathsf{ShortPath}_G(T, \mu) = \{\pi \in \mathsf{Plays}(G) \mid \mathsf{TS}_T(\pi) \leq \mu\}\,.$$

Notice the inequality is reversed (lower is better).

As all weights are strictly positive, it is possible to reduce the truncated sum to the total-payoff (i.e., for all $\pi \in \mathsf{Plays}(G)$, $\mathsf{TS}_T(\pi) = \mathsf{TP}(\pi)$) by making all states of T absorbing with a self-loop of zero weight. That is, for all $s \in T$, we have that $\mathsf{Succ}(s) = \{s\}$ and $w((s, s)) = 0$.

However, this problem is simpler than the total-payoff. Both for games and MDPs, pure memoryless strategies suffice. Moreover, both threshold problems can be solved in polynomial time. In games (worst-case semantics), winning strategies of $\mathcal{P}_1$ should avoid all cycles (because they yield strictly positive costs). Hence usage of attractors and comparison of the worst possible sum of costs with the threshold suffices. In MDPs (expected value semantics), the shortest path problem is solvable in polynomial time via linear programming [BT91, dA99].

Energy. The *energy* objective aims at modeling resource consumption and possible exhaustion over an infinite play. The goal of $\mathcal{P}_1$ is thus to maintain

the running sum of weights (i.e., the total-payoff over prefixes) above zero at all times. To stress on the context of energy games, we introduce the *energy level* of a sequence of states $\rho = s_0 s_1 \dots s_n$, which is simply is $\mathsf{EL}(\rho) = \mathsf{TP}(\rho)$. This alternative notation is used to easily distinguish between the two contexts (as the associated objectives will differ) and improve readability.

The usual question for the energy objective is slightly different than for the preceding ones. We are interested in the *unknown initial credit problem*. That is: does there exist a finite initial credit $v_0 \in \mathbb{N}$ and a strategy $\lambda_1 \in \Lambda_1$ of $\mathcal{P}_1$ such that $\mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1) \subseteq \mathsf{Energy}_G(v_0)$, for

$$\mathsf{Energy}_G(v_0) = \{\pi \in \mathsf{Plays}(G) \mid \forall\, n \geq 0 : v_0 + \mathsf{EL}(\pi(n)) \in \mathbb{N}\}\,.$$

This objective is essentially a safety objective over the running sum of weights. Consequently, the sure semantics is the one of choice for the analysis of this quantitative setting. A discussion of the inadequacy of almost-sure semantics in this context is presented in Sect. 7.2. Note that expectation analysis is not relevant as there is no value function to optimize.

In one-dimension games, pure memoryless strategies suffice for the energy objective and the unknown initial credit problem is in NP $\cap$ coNP [CdAHS03]. Morever, it was proved by Bouyer et al. that energy games are log-space equivalent to mean-payoff games [BFL+08]. Specifically, given a game G, $\mathcal{P}_1$ has a strategy and a finite initial credit to win the energy objective if and only if he has a strategy to ensure a non-negative mean-payoff.

Example 2.28. Let us consider the game in Fig. 2.1. The equivalence between mean-payoff games and energy games can be observed. Indeed, notice that the strategy consisting in always taking left has been shown to ensure a mean-payoff of zero. Hence, this strategy permits to win the energy objective for some finite initial credit, which in this case is 1 (to be able to consume one unit of energy when taking the -1 edge for the first time).

We mentioned that almost-sure semantics is inadequate for energy objectives. In particular, it is not more powerful than sure semantics. To illustrate this fact, consider the MC in Fig. 2.4. Whatever the finite initial credit, the energy objective cannot be ensured surely because of the play $(s_1)^\omega$ which requires an infinite amount of resources. Now assume we relax the objective by considering

the almost-sure semantics, as we know that this play $(s_1)^\omega$ has probability zero. Despite that, for any initial credit $v_0 \in \mathbb{N}$, there is a set of plays $(s_1)^{v_0+1}(s_1)^*(s_2)^\omega$ which has a strictly positive probability measure and such that all such plays are losing for the energy objective. Intuitively, we see that finite witnesses of energy exhaustion always exist if surely winning cannot be ensured. Such prefixes have a non-zero probability and thus almost-surely winning is also hopeless. This argument is presented formally in Sect. 7.2. ◁

CHAPTER 3

Contributions

Multi-Dimension Objectives ◇ Window Objectives ◇ Beyond Worst-Case Synthesis

---

Within this chapter, we give a brief presentation of the questions addressed in the thesis and we summarize our main contributions. It is intended to be an entry point to our work.

Our aim is to provide the reader with an overview of our core results, through concrete statements. Technical details, as well as additional results, are spread over Parts II, III and IV.

3.1 Multi-Dimension Objectives

In collaboration with Krishnendu Chatterjee (Institute of Science and Technology Austria) and Jean-François Raskin (Université Libre de Bruxelles, Belgium) [CRR12a, CRR12b, CRR13], later also involving Laurent Doyen (LSV - ENS Cachan, France) [CDRR13a, CDRR13b], we studied extension of classical quantitative objectives to multi-dimension games. In such games, edges bear vectors of weights and traditional objectives are naturally extended. Table 4.1 presents an overview of the situation for multi-dimension games compared to one-dimension games.

The worst-case mean-payoff threshold problem and the unknown initial credit problem for energy games are both coNP-complete in the multi-dimension setting [CDHR10, VR11, VCD+12]. We review the situation for such games in Sect. 4.2.1. Our first question considers the total-payoff objective, for which no result was known.

Question 1. *Given that mean-payoff and total-payoff games are closely related in the one-dimension case (Lemma 4.7), and they both are in $UP \cap coUP$, does this relation extend to the multi-dimension case? In particular, do multi total-payoff games belong to coNP?*

▷ We prove that total-payoff games are undecidable in multi-dimension games with at least five dimensions (Thm. 4.8), even when $\mathcal{P}_1$ is restricted to finite-memory strategies (Sect. 4.3.2). Our proof uses a reduction from the halting problem for two-counter machines [Min61].

For multi-dimension mean-payoff games, infinite-memory strategies are more powerful than finite-memory ones, which is not the case for energy games. Moreover, both objectives are equivalent when restricted to finite-memory strategies [CDHR10, VR11, VCD+12]. The finite-memory model is the one of choice for controller synthesis (Sect. 2.2.3). Therefore, it is crucial to precisely characterize the memory requirements. We studied this problem for multi-dimension energy (and mean-payoff) in conjunction with parity.

Question 2. *Can we bound the memory needed for winning strategies in multi-energy parity games?*

▷ For $\mathcal{P}_1$, we prove that single exponential memory is sufficient (Thm. 5.10).

Our result is a significant improvement over the triple exponential upper bound that can be obtained naively from the results known in literature [BJK10]. Our approach relies on the notion of *even-parity self-covering trees* representing winning strategies, and is in two steps. First, we bound the depth of such trees by a refined analysis (Lemma 5.4). Second, we transform them in directed acyclic graphs of single exponential width by merging comparable nodes (Lemma 5.6).

- ▷ For $\mathcal{P}_1$, we also show that exponential memory is necessary even when all weights belong to $\{-1, 0, +1\}$ and with no parity condition (Lemma 5.9).
- ▷ For $\mathcal{P}_2$, pure memoryless strategies are sufficient (Lemma 5.1).

Deciding if a winning strategy exists is in coNP, but what we want is to actually synthesize such a winning strategy.

Question 3. *Can we obtain an efficient synthesis algorithm for multi-dimension energy (or mean-payoff) parity games?*

- ▷ We establish a synthesis algorithm that requires exponential time in the worst-case (Thm. 6.8). Since exponential memory is required (and the decision problem is coNP-complete), this worst case exponential bound can be considered optimal.
- ▷ Our algorithm is *symbolic* in the sense that it uses a compact antichain representation of sets of winning credits by their minimal elements [DDHR06]. It is also *incremental*: we search for winning credits within a small range, and increment the range only when necessary. This ensures efficient implementation in practice.

Finally, we answer the following question for several classes of games. Our results are summarized in Table 7.1.

Question 4. *Can we trade pure finite-memory strategies of $\mathcal{P}_1$ for conceptually simpler randomized memoryless strategies, if we relax from sure semantics to almost-sure semantics or expectation semantics?*

- ▷ For energy objectives, the answer is always no. Randomization is not helpful even with only one player, as energy objectives are similar in spirit to safety objectives (Lemma 7.2).

▷ For two-player multi-dimension mean-payoff games, the answer is no, even without parity. The answer becomes yes for one-player games, even with parity (Lemma 7.4).

▷ For two-player games with a single mean-payoff objective and a parity objective, the answer is yes (Lemma 7.10).

3.2 Window Objectives

While studying the mean-payoff and total-payoff objectives, it became clear for us that their applicability can prove to be challenging for at least two reasons. First, there is the obvious question of *tractabity*. Both objectives face a long-standing complexity barrier in one-dimension games as no polynomial-time algorithm is known (Sect. 2.3.2). In multi-dimension games, we discussed that total-payoff is out of the picture due to undecidability. Second, both objectives are not well-suited for systems where some (discrete) notion of time is important. Indeed, they characterize *long-run* behaviors over infinite plays whereas in many practical problems, we wish to provide bounds on *time frames* in which an acceptable behavior can be witnessed.

Through joint work with Krishnendu Chatterjee (Institute of Science and Technology Austria), Laurent Doyen (LSV - ENS Cachan, France) and Jean-François Raskin (Université Libre de Bruxelles, Belgium) [CDRR13a,CDRR13b], we introduced novel quantitative objectives, named *window objectives*. They partially answer the issues raised by classical mean-payoff and total-payoff.

Our window objectives strengthen classical total-payoff and mean-payoff with timing guarantees. Instead of the (total- or mean-)payoff along the whole infinite play, we consider payoffs over a local bounded window sliding along the play. We study several variants (Sect. 8.2), the main ones being the *bounded window mean-payoff* and *fixed window mean-payoff* objectives. In the fixed window variant, the bound on the window size is given as a parameter. In the bounded window variant, the question is whether there exists a finite bound such that the corresponding fixed window objective is satisfied on the considered play.

The first natural question is whether those objectives are related to the classical ones, and how.

Question 1. *What is the relation between window objectives and classical mean-payoff and total-payoff?*

- ▷ We prove that the window mean-payoff is a *conservative approximation* of the classical mean-payoff in the following sense (Lemma 8.3): if the (fixed or bounded) window objective is won for threshold zero, then the mean-payoff objective is also won for threshold zero; if the mean-payoff objective is won for threshold strictly greater than zero, then the bounded window objective is won for threshold zero.
- ▷ Similar results are obtained for the total-payoff with variants of the window objective (also Lemma 8.3).

Based on that result, it seemed promising to investigate the complexity of window objectives, as they could provide a more tractable alternative to their classical counterparts. We started with the one-dimension setting (comparative overview in Table 8.1).

Question 2. *What are the complexities of decision problems for the different variants in one-dimension games? What are the memory requirements?*

- ▷ One-dimension fixed window games are solvable in time *polynomial* in the size of the game and in the maximal window size (Thm. 9.6) via a recursive algorithm. This implies that the problem is P-complete for polynomial window sizes, in contrast to the long-standing UP ∩ coUP barrier for classical objectives. Memory is necessary for both players and linear memory always suffice.
- ▷ For the bounded window, we prove that the problem is in NP ∩ coNP and at least as hard as classical mean-payoff games (Thm. 9.16). Moreover, we show that $\mathcal{P}_1$ can always use memoryless strategies whereas $\mathcal{P}_2$ may require infinite memory in general. This is an interesting reversal of the situation for classical quantitative objectives where $\mathcal{P}_2$ is usually memoryless, even in multi-dimension games or conjunctions with parity (Table 2.2, Table 4.1 and Table 4.2).

Lastly, we extended our study of window objectives to multi-dimension games (comparative overview in Table 8.2).

Question 3. *What are the complexities of decision problems for the different variants in multi-dimension games? What are the memory requirements?*

- ▷ For the fixed window, we give several complexity results (Thm. 10.9). For *arbitrary* window sizes, we prove the problem to be EXPTIME-complete. Membership follows from a reduction to exponentially larger co-Büchi games (Lemma 10.2). Hardness is proved in two settings: arbitrary dimensions and weights in $\{-1, 0, 1\}$ via a reduction from the membership problem in alternating polynomial-space Turing machines [CKS81] (Lemma 10.5), and only two dimensions with arbitrary weights by reduction from countdown games [JSL08] (Lemma 10.6). For *polynomial* windows, we get PSPACE-hardness via generalized reachability games [FH10] (Lemma 10.7). Finally, in both cases, exponential memory is both sufficient and necessary (Lemma 10.8).
- ▷ Unfortunately, we establish a prohibitive lower bound for the complexity of bounded window games: they are at least non-primitive recursive. We prove it by reduction from the termination problem in reset nets [Sch02, LNO+08] (Thm. 10.10). Decidability remains open.

3.3 Beyond Worst-Case Synthesis

Our last set of results is related to the new framework of *beyond worst-case synthesis* (Def. 11.3). Given a two-player game, a value function and finite-memory stochastic model for $\mathcal{P}_2$ (given as a stochastic output Moore machine), we essentially look for strategies that combine satisfaction of a worst-case problem in the two-player game and satisfaction of an expected value threshold problem in the MDP induced by the stochastic model. This model was introduced and studied in collaboration with Véronique Bruyère (Université de Mons, Belgium), Emmanuel Filiot and Jean-François Raskin (both from Université Libre de Bruxelles, Belgium) [BFRR13, BFRR14a, BFRR14b].

This novel problem aims at mixing high expected performance with strict worst-case guarantees, a specification that is not expressible in existing models (Sect. 11.1.1). We focused on synthesizing finite-memory strategies and we only studied one-dimension games. We first studied the problem for the mean-payoff (comparative overview in Table 11.1).

Question 1. *What is the complexity of the beyond worst-case mean-payoff problem? What are the memory requirements?*

- ▷ We establish an algorithm in NP ∩ coNP. It is based on refined analysis of end-components and combination of simple strategies into more complex combined strategies (Thm. 12.1). Our algorithm is optimal with regard to the complexity of the worst-case mean-payoff threshold problem which is already not known to be in P. Hence we interestingly maintain the *same tractability level while increasing the expressive power*.
- ▷ Memory of pseudo-polynomial size (polynomial in the size of the game and in the values of weights and thresholds) is sufficient and in general necessary for finite-memory strategies of $\mathcal{P}_1$ (Thm. 12.35). For $\mathcal{P}_2$, memoryless strategies suffice as the objective is essentially a disjunction of the worst-case and expected value threshold problems (for which memoryless optimal strategies exist as discussed in Sect. 2.3.2).
- ▷ Infinite-memory strategies are strictly more powerful than finite-memory ones for $\mathcal{P}_1$. However they are less useful for synthesis and their analysis is technically challenging (Sect. 12.7).

We also studied the beyond worst-case problem for the shortest path (comparative overview in Table 11.2).

Question 2. *What is the complexity of the beyond worst-case shortest path problem? What are the memory requirements?*

- ▷ We establish a pseudo-polynomial-time algorithm based on a sequential approach, first taking care of the worst-case, then of the expected value (Thm. 13.3). Such an approach is conceptually simpler than in the mean-payoff case but cannot be transposed for the mean-payoff (Rem. 13.1).
- ▷ In contrast to the mean-payoff case, our beyond worst-case algorithm shows a complexity leap with regard to the individual worst-case and expected value threshold problems which are in P for the shortest path. It cannot be avoided unless P = NP as the problem is NP-hard (Thm. 13.6). We use a reduction from the K^{th} largest subset problem to prove it [JK78, GJ79].
- ▷ Whereas infinite memory is useful in the beyond worst-case mean-payoff setting, we show that for the shortest path, pseudo-polynomial memory is

always sufficient for $\mathcal{P}_1$. It is also necessary in general (Thm. 13.4). For $\mathcal{P}_2$, memoryless strategies again suffice as it is the case in the corresponding worst-case and expected value threshold problems (Sect. 2.3.2).

Part II

Multi-Dimension Objectives

CHAPTER 4

Multi-Dimension Quantitative Objectives

Introduction ◇ Pure Strategies ◇ Pure Finite-Memory Strategies

We open Part II with motivation regarding the analysis of games with multi-dimension objectives. This framework, as well as conjunction of quantitative and qualitative objectives, has many applications in verification and synthesis.

We then focus on decision problems in multi-dimension mean-payoff or energy games, based on recent results by Velner et al. [VCD+12] showing coNP-completeness. We prove that albeit closely related to mean-payoff objectives in one-dimension games, total-payoff objectives become undecidable in multi-dimension games. This result was published in [CDRR13a, CDRR13b].

Multi-dimension mean-payoff requires infinite memory for the first player. We are interested in synthesizing finite-memory controllers. Hence we restrict the first player to finite-memory strategies and present corresponding results.

Finally, we discuss one-dimension mean-payoff or energy combined with parity [CHJ05, BMOU11, CD12]. We advocate conjunction of parity and multi-dimension objectives, for which we present first results in the following chapters.

4.1 Introduction

Multi-dimension objectives. In applications of verification and synthesis, quantitative objectives that typically arise are (i) multi-dimension quantitative objectives (i.e., conjunction of several quantitative objectives), e.g., to express properties like the average response time between a grant and a request is below a given threshold μ_1, and the average number of unnecessary grants is below threshold μ_2; and (ii) conjunction of quantitative objectives with a Boolean objective, such as a mean-payoff parity objective that can express properties like the average response time is below a threshold along with satisfying a liveness property.

In summary, the quantitative objectives can express properties related to resource requirements, performance, and robustness; multiple objectives can express the different, potentially dependent or conflicting objectives; and the Boolean objective specifies functional properties such as liveness or fairness.

The framework of multi-dimension quantitative games and games with conjunction of quantitative and Boolean objectives has recently been shown to have many applications in verification and synthesis, such as synthesizing systems with quality guarantee [BCHJ09], synthesizing robust systems [BGHJ09], performance aware synthesis of concurrent data structure [CCH+11], analyzing permissivity in games and synthesis [BMOU11], simulation between quantitative automata [CDH10a], generalizing Boolean simulation to quantitative simulation distance [CHR12], etc. Moreover, multi-dimension energy games are equivalent to a decidable class of games on *vector addition systems with states* (VASS). This model is equivalent to games over multi-counter systems and Petri nets [BJK10]. Various decision problems over multi-dimension energy games were studied in [FJLS11].

There are many recent works on the theoretical analysis of multi-dimension quantitative games, such as, mean-payoff parity games [CHJ05, BMOU11], energy-parity games [CD12], multi-dimension energy games [CDHR10, VCD+12], and multi-dimension mean-payoff games [CDHR10, VR11, VCD+12].

Synthesis. Most of these works focus on establishing the computational complexity of the problem of deciding if player 1 *has* a winning strategy. From the perspective of synthesis and other related problems in verification, the most im-

		EG	$\underline{\mathsf{MP}}$	$\overline{\mathsf{MP}}$	$\underline{\mathsf{TP}}$	$\overline{\mathsf{TP}}$
one-dim.	complexity	NP ∩ coNP				
	$\mathcal{P}_1$ mem.	pure memoryless				
	$\mathcal{P}_2$ mem.					
k-dim.	complexity	coNP-c.		NP ∩ coNP	**undec.**	
	$\mathcal{P}_1$ mem.	pure finite	pure infinite		-	
	$\mathcal{P}_2$ mem.	pure memoryless				

Table 4.1: Overview of results for multi-dimension quantitative objectives. New results are in bold.

portant problem is to obtain a witness *finite-memory* winning strategy (if one exists), as discussed in Sect. 2.2.3. The winning strategy in the game corresponds to the desired controller for (or implementation of) the system in synthesis, and for implementability a finite-memory strategy is essential.

Our contributions. Within Part II, we consider the problem of finite-memory strategy synthesis in multi-dimension quantitative games in conjunction with parity objectives, and the problem of existence of memory-efficient randomized strategies for such games.

This chapter discusses complexity of decision problems in games with multi-dimension quantitative objectives, and memory requirements for winning strategies. An overview of the results for games without parity is presented in Table 4.1. One-dimension games with parity are discussed in Sect. 4.3.3 and an overview is given in Table 4.2.

We are the first to consider the conjunction of multi-dimension quantitative objectives with a parity condition: our results are presented in the following chapters. They were obtained through collaboration with Chatterjee and Raskin [CRR12a, CRR12b, CRR13]. In this chapter, we also discuss undecidability of multi-dimension total-payoff objectives, proved jointly with Chatterjee, Doyen and Raskin [CDRR13a, CDRR13b].

4.1.1 Assumptions and Additional Notations

Multi-dimension games. Throughout Chap. 4 to 7, we specifically address the setting of games with *multi-dimension weight functions*. That is, $w\colon E \to \mathbb{Z}^k$ where $k \in \mathbb{N}$ denotes the *dimension* of the weight vectors. Similarly, we consider multi-dimension value functions $f\colon \mathsf{Plays}(G) \to (\mathbb{R} \cup \{-\infty, \infty\})^k$.

Therefore, we generalize our game notation to $G = (S_1, S_2, E, k, w)$, to include the number of dimensions explicitly.

Generalized objectives. The quantitative objectives defined in Sect. 2.3 are naturally generalized to the multi-dimension context. Whenever limits are involved, we consider componentwise limits. For the energy objective, the energy level must stay positive in all dimensions at all times. Note that we sometimes simply write *multi* instead of *multi-dimension* for the sake of brevity.

Pareto optimality. In the multi-dimension setting, we have to deal with vectors of integer weights. The usual order on those vectors is partial. For example, if no additional preference is given, vector $(1, 2)$ is incomparable with vector $(2, 1)$. Therefore, the notion of optimal value is ill-defined, and it is natural to consider so-called *Pareto-optimal* values. An element a of a set $A \subseteq \mathbb{Z}^k$ is Pareto-optimal (for the partial order $\geq$) in A if for all $a' \in A \setminus \{a\}$, there exists a dimension l, $1 \leq l \leq k$ such that $a'(l) < a(l)$.

Games with a parity objective. To improve readability of the following chapters, we introduce a different notation for games extended with a priority function $p\colon S \to \mathbb{N}$. We denote them as $G_p = (S_1, S_2, E, k, w, p)$. Observe that any game G without parity can trivially be seen as a game G_p with parity by considering priority zero for all states.

Pure or general strategies. Within Chap. 4 through 6, we mostly study the worst-case threshold problem for the mean-payoff objective and the unknown initial credit problem for the energy one. Both are considered using the *sure semantics*. Consequently, randomization adds no power and we focus on *pure strategies*, which are sufficient to win.

Nonetheless, in Chap. 7, we investigate the particular case of pure finite-memory strategies and study if introducing randomization can help in reducing the memory requirements. It is indeed the case for some objectives, where randomization can actually replace finite memory in winning strategies.

4.2 Pure Strategies

4.2.1 Mean-Payoff and Energy

Multi-dimension mean-payoff and energy games (without parity) have been recently introduced by Chatterjee et al. under the name of *generalized* mean-payoff and energy games [CDHR10]. Complexities of the decision problems for finite-memory strategies were investigated. It was also noted that infinite memory is strictly more powerful for multi-dimension mean-payoff games. However, complexity for infinite-memory strategies was left open. In [VR11], Velner and Rabinovich solved this open question. Our presentation here is mainly based on a joint paper by both groups of authors that gives a complete study of the problem [VCD+12]. We sketch some of the important results and notions necessary to understand our following contributions.

Multi energy games. Finite-memory strategies are always sufficient in multi-dimension energy games.

Lemma 4.1 ([VCD+12, Lemma 1 and 2]). *If $\mathcal{P}_1$ has a winning strategy in a multi energy game, then he has a pure finite-memory winning strategy. If $\mathcal{P}_2$ has a winning strategy, then he has a pure memoryless winning strategy.*

Proof sketch. First consider the case of $\mathcal{P}_1$. Assume $\mathcal{P}_1$ has a pure (possibly infinite-memory) strategy $\lambda_1 \in \Lambda_1^P$ that ensures the energy objective for some initial credit $v_0 \in \mathbb{N}^k$. The goal is to build a pure finite-memory strategy $\lambda_1^f \in \Lambda_1^{PF}$ that is also winning for the same initial credit.

To achieve this, consider the unfolding of the game when strategy λ_1 is used by $\mathcal{P}_1$. This gives an infinite tree. Each node in this tree can be labeled by its corresponding energy level, representing the accumulated energy along the path from the root down to this node, starting from the initial credit level. Clearly, all nodes must be labeled with vectors of naturals since λ_1 is winning. One can define a relation on the set $S \times \mathbb{N}^k$ based on the fact that $\mathcal{P}_1$ has more flexibility when he has an higher amount of accumulated energy. That is, $(s_1, v_1) \leq (s_2, v_2)$ if $s_1 = s_2$ and $v_1 \leq v_2$. This relation is a well-quasi-order: it is transitive, reflexive and has no infinite descending sequence nor infinite antichain (set of incomparable elements).

Consequently, any infinite branch (each representing a play) in the tree admits two positions $i < j$ such that the energy level has increased between two visits of the same state: $(s_i, v_i) \leq (s_j, v_j)$. Based on this observation, one can stop each branch when such a position j is reached. It can be proved that the resulting tree is finite, thanks to Kőnig's lemma [Kő36] and Dickson's lemma [Dic13]. When in position j, $\mathcal{P}_1$ can essentially reenact the same behavior as he did from i to j: if he was able to maintain a positive energy starting from level v_i, he will also be able to do it starting from $v_j \geq v_i$. Hence from this finite tree, it is possible to extract a corresponding finite-memory winning strategy, for initial credit v_0.

Sufficiency of pure memoryless strategies for $\mathcal{P}_2$ follows from results on VASS games [BJK10, Lemma 19]. A key idea of the argument is the following. Let $s \in S_2$ be a state in which $\mathcal{P}_2$ can choose between two successors s' and s'' such that $\mathcal{P}_1$ can win from s' with initial credit v' and from s'' with initial credit v''. Then, $\mathcal{P}_1$ can ensure winning from s with initial credit $v = v' + v''$. Consequently, $\mathcal{P}_2$ cannot benefit from alternating between s' and s'' (in the sense that he still cannot win using memory if he is not able to win without memory). The combination of winning credits for s' and s'' into a winning credit for s is part of the synthesis algorithm developed in Chap. 6, as illustrated in Fig. 6.1. □

Remark 4.2. Memory is useless for $\mathcal{P}_2$ in the sense that it never modifies the answer to the unknown initial credit problem. Nevertheless, it may have some impact on the required initial credit to win when $\mathcal{P}_1$ has a winning strategy. We depict this case in Fig. 4.1 (presented in [VCD+12]). When $\mathcal{P}_2$ is memoryless, initial credit $(2, 0)$ suffices whether $\mathcal{P}_2$ decides to go left or right. In the latter case, $\mathcal{P}_1$ has to alternate between the upper and lower edges, starting with the latter. Observe that $\mathcal{P}_1$ needs memory to do so. Now if $\mathcal{P}_2$ has memory, he can first go right once then go left. In that case, $\mathcal{P}_1$ needs either an initial credit $(2, 1)$ (if he comes back with the upper edge), or an initial credit $(3, 0)$ (if he takes the lower edge) to win. ◁

We now present precise complexity of the decision problem.

Theorem 4.3 ([VCD+12, Thm. 3]). *The unknown initial credit problem in multi energy games is coNP-complete.*

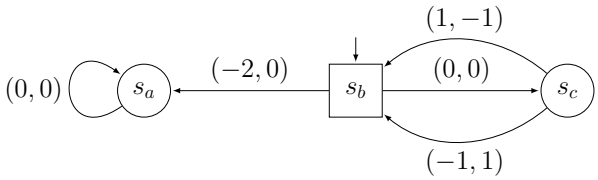


Figure 4.1: Memory of $\mathcal{P}_2$ has an impact on needed initial credit: $(2, 0)$ suffices when $\mathcal{P}_2$ is memoryless, not when he uses memory.

Key ideas. Thanks to memoryless strategies being sufficient for $\mathcal{P}_2$, the unknown initial credit problem in two-player games can be shown to be in coNP if the one-player version (with only $\mathcal{P}_1$) is proved to be in P. Indeed, it then suffices to guess a memoryless strategy of $\mathcal{P}_2$ and check that it is winning by solving the one-player problem.

A polynomial algorithm for the one-player game, which is basically a graph problem, is obtained by reduction to the problem of checking the existence of a zero circuit in the graph. That is, a not necessarily simple cycle[1] with non-negative energy level in all dimensions. This problem is known to be in P [KS88].

The hardness result is by reduction from the complement of the 3SAT problem, which is coNP-complete [Pap94]. □

Multi mean-payoff games. In contrast to multi energy games, infinite memory may be needed to win multi mean-payoff games in general. This is already the case in one-player games. For $\mathcal{P}_2$, proving that memoryless strategies are sufficient requires different approaches for the infimum and supremum variants. In the supremum case, it is based on an induction argument on the number of states and existence of memoryless winning strategies of $\mathcal{P}_2$ on games projected to individual dimensions. In the infimum case, it uses a general result by Kopczynski granting existence of memoryless strategies for $\mathcal{P}_2$ in all games with objectives that are convex (i.e., closed under plays combination) and prefix-independent [Kop06]. The infimum mean-payoff satisfies these hypotheses.

Lemma 4.4 ([VCD+12, Thm. 6 and 7]). *In general, infinite-memory strategies are necessary for $\mathcal{P}_1$ in multi mean-payoff games, both for infimum and supremum variants. In both cases, memoryless strategies suffice for $\mathcal{P}_2$.*

[1] A cycle is *simple* if it contains no repeated state, except for the origin state of course.

To illustrate the need for infinite memory, we consider the following example, given in [VCD+12].

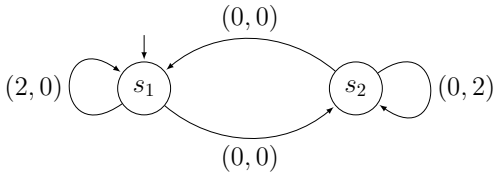


Figure 4.2: Infinite memory is needed to win, both for objectives $\mathsf{MeanInf}_G((1,1))$ and $\mathsf{MeanSup}_G((2,2))$.

Example 4.5. Consider the one-player game depicted in Fig. 4.2.

First, let us study what kind of thresholds can be ensured by finite-memory strategies. Note that such strategies induce *ultimately periodic plays*.[2] Thus, infimum and supremum limits coincide. Let us fix some pure finite-memory strategy $\lambda_1^f \in \Lambda_1^{PF}$. It induces a unique consistent play. Two cases are possible: either it visits infinitely often both states, or it eventually settles on one of them. In the former case, the middle edges appear in the periodic part. Hence, the mean-payoff is of the form $a \cdot (2,0) + b \cdot (0,0) + c \cdot (0,2)$, with $a, b, c \in]0,1[\cap \mathbb{Q}$ and $a + b + c = 1$. In the latter case, only vectors $(2,0)$ and $(0,2)$ can be obtained. Consequently, threshold vectors $(1,1)$ and a fortiori $(2,2)$ cannot be achieved with a finite-memory strategy.

Second, focus on the infimum mean-payoff objective, with threshold $(1,1)$. It is possible to achieve this threshold using infinite memory. Indeed, let λ_1 be the strategy that loops $n \in \mathbb{N}$ times on s_1, then goes to s_2 and loops n times on s_2 before going back to s_1, and then repeats this process forever for increasing values of n. In the long-run, the contribution of the middle edges tends to zero, and the infimum mean-payoff is $(1,1)$. Observe that infinite memory is necessary to implement this strategy as n needs to grow boundlessly.

Finally, consider the supremum variant of the objective and a strategy that is also based on alternating between s_1 and s_2 and looping on these states for longer and longer times. However, instead of looping equal times on both states, this strategy loops for longer and longer times each time it changes state. In

[2] A play $\pi \in \mathsf{Plays}(G)$ is *ultimately periodic* if it can be decomposed as $\pi = \rho \cdot (\rho')^\omega$, for some finite prefixes $\rho, \rho' \in \mathsf{Prefs}(G)$.

particular, by increasing the number of loops sufficiently, we can ensure that the contribution of the preceding finite prefix becomes negligible. In the long-run, the supremum mean-payoff is equal to $(2,0)+(0,2)=(2,2)$.

Notice the gap existing between thresholds achievable for both variants when infinite-memory strategies are considered. Clearly, from a practical standpoint, threshold $(2,2)$ is unrealistic for actual controllers. This observation further indicates that the infinite-memory model is inadequate for $\mathcal{P}_1$ in terms of applicability. ◁

Complexity of the threshold problem when infinite memory is allowed varies slightly depending on the considered variant.

Theorem 4.6 ([VCD+12, Thm. 6 and 7]). *The worst-case threshold problem is in NP∩coNP for multi supremum mean-payoff games and coNP-complete for multi infimum mean-payoff games.*

Key ideas. The NP ∩ coNP-membership of the supremum variant relies on a reduction to polynomially many one-dimension supremum mean-payoff games. The coNP-hardness result for multi infimum mean-payoff games is based on a reduction from the complement of 3SAT, similar to the one used for multi energy games. The coNP-membership follows from sufficiency of memoryless strategies for $\mathcal{P}_2$ and P-membership of the one-player problem. This one-player version is solved by reduction to a variant of the zero circuit problem that was used for multi energy games. This variant is called the non-negative multi-cycle problem. The difference is that involved cycles do not need to be connected in this variant. Intuitively, the potential negative effect induced by switching from cycles to cycles (e.g., using the middle edges in Fig. 4.2) can be made negligible by having the frequency of switching tending to zero. Hence it can be ignored for the mean-payoff. Note that this is not the case for energy objectives, as such connecting edges may decrease the energy. Even if their frequency tends to zero, they are still used infinitely often and may prevent the energy objective from being satisfied. □

4.2.2 Total-Payoff

In Sect. 4.2.1, we surveyed the core results on multi-dimension mean-payoff and energy games. Interestingly, the case of multi-dimension total-payoff games

was unexplored prior to this thesis and the corresponding publications. In this section we present results obtained jointly with Chatterjee, Doyen and Raskin [CDRR13a, CDRR13b].

We first discuss the close relation existing between mean-payoff and total-payoff in one-dimension games, and show that this relation breaks in multiple dimensions. We then establish that the worst-case threshold problem for total-payoff surprisingly becomes undecidable in the multi-dimension setting, both for the infimum and supremum variants.

Total-payoff vs. mean-payoff. First, consider one-dimension games. In this case, memoryless strategies exist for both players for both objectives, as discussed in Sect. 2.3.2 [LL69, EM79, FV97, GZ04] and the supremum and infimum mean-payoff problems coincide (which is not the case for total-payoff). Worst-case threshold problems for mean-payoff and total-payoff are closely related as witnessed by Lemma 4.7 and both have been shown to be in NP∩coNP [ZP96, GS09] (and even in UP∩coUP [Jur98, GS09]). We denote by $\{0\}^k$ the k-dimension zero vector.

Lemma 4.7. *Let $G = (S_1, S_2, E, k, w)$ be a two-player game and $s_{\mathsf{init}} \in S$ be an initial state. Let A, B, C and D respectively denote the following assertions.*

A. Player $\mathcal{P}_1$ has a winning strategy for $\mathsf{MeanSup}_G(\{0\}^k)$.

B. Player $\mathcal{P}_1$ has a winning strategy for $\mathsf{MeanInf}_G(\{0\}^k)$.

C. There exists a threshold $\mu \in \mathbb{Q}^k$ such that $\mathcal{P}_1$ has a winning strategy for $\mathsf{TotalInf}_G(\mu)$.

D. There exists a threshold $\mu' \in \mathbb{Q}^k$ such that $\mathcal{P}_1$ has a winning strategy for $\mathsf{TotalSup}_G(\mu')$.

For games with one-dimension ($k = 1$) weights, all four assertions are equivalent. For games with multi-dimension ($k > 1$) weights, the only implications that hold are: $C \Rightarrow D \Rightarrow A$ and $C \Rightarrow B \Rightarrow A$. All other implications are false.

The statement of Lemma 4.7 is depicted in Fig. 4.3: the only implications that extend to the multi-dimension case are depicted by solid arrows.

Proof. Specifically, the implications that remain true in multi-dimension games are the trivial ones: satifaction of the infimum version of a given objective trivially implies satisfaction of its supremum version, and satisfaction of infimum

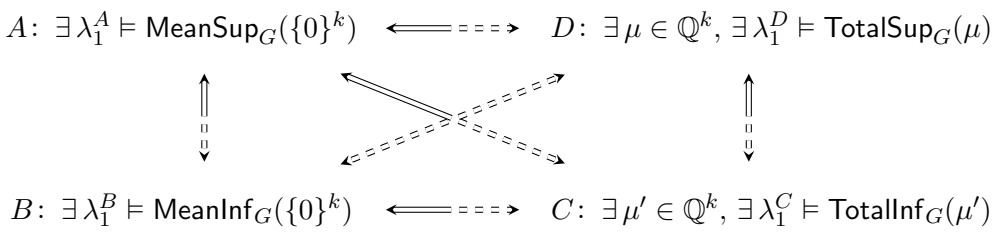


Figure 4.3: Equivalence between worst-case threshold problems for mean-payoff and total-payoff. Dashed implications are only valid for one-dimension games.

(resp. supremum) total-payoff for some finite threshold $\mu \in \mathbb{Q}^k$ implies satisfaction of infimum (resp. supremum) mean-payoff for threshold $\{0\}^k$ as from some point on, the corresponding sequence of mean-payoff infima (resp. suprema) in all dimensions t, $1 \leq t \leq k$, can be lower-bounded by a sequence of elements of the form $\frac{\mu(t)}{n}$ with n the length of the prefix, which tends to zero for an infinite play. That is thanks to the sequence of total-payoffs over prefixes being a sequence of integers: it always achieves the value of its limit $\mu(t)$ instead of only tending to it asymptotically as could a sequence of rationals such as the mean-payoffs. This sums up to $C \Rightarrow D \Rightarrow A$ and $C \Rightarrow B \Rightarrow A$ being true even in the multi-dimension setting.

In the one-dimension case, all assertions are equivalent. First, we have that infimum and supremum mean-payoff problems coincide as memoryless strategies suffice for both players. Thus, we add $A \Rightarrow B$ and $D \Rightarrow B$ by transitivity. Second, consider an optimal strategy for $\mathcal{P}_1$ for the mean-payoff objective of threshold 0. This strategy is such that all cycles formed in the outcome have non-negative effect, otherwise $\mathcal{P}_1$ cannot ensure winning. Thus, the total-payoff over any outcome that is consistent with the same optimal strategy is at all times bounded from below by $-2 \cdot (|S| - 1) \cdot W$ (once for the initial cycle-free prefix, and once for the current cycle being formed). Therefore, we have that $B \Rightarrow C$, and we obtain all other implications by transitive closure.

For multi-dimension games, all dashed implications are false. We specifically consider two of them.

1. To show that implication $D \Rightarrow B$ does not hold, consider the one-player game depicted in Fig. 4.4. Clearly, any finite vector $\mu \in \mathbb{Q}^k$ for the supremum total-payoff objective can be achieved by an infinite memory strategy

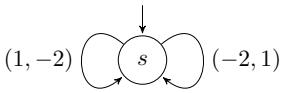


Figure 4.4: Satisfaction of supremum total-payoff does not imply satisfaction of infimum mean-payoff.

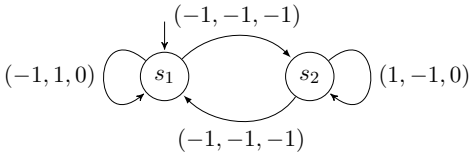


Figure 4.5: Satisfaction of infimum mean-payoff does not imply satisfaction of supremum total-payoff.

consisting in playing both loops successively for longer and longer periods, each time switching after getting back above the threshold in the considered dimension. However, it is impossible to build any strategy, even with infinite memory, that provides an infimum mean-payoff of $(0, 0)$ as the limit mean-payoff would be at best a linear combination of the two cycles values, i.e., strictly less than zero in at least one dimension in any case.

2. Lastly, failure of implication $B \Rightarrow D$ in multi-dimension games can be witnessed in Fig. 4.5. Clearly, the strategy that plays for n steps in the left cycle, then goes for n steps in the right one, then repeats for $n' > n$ and so on, is a winning strategy for the infimum mean-payoff objective of threshold $(0, 0, 0)$. Nevertheless, for any strategy of $\mathcal{P}_1$, the outcome is such that either (i) it only switches between cycles a finite number of time, in which case the sum in dimension 1 or 2 will decrease to infinity from some point on, or (ii) it switches infinitely and the sum of weights in dimension 3 decreases to infinity. In both cases, the supremum total-payoff objective is not satisfied for any finite vector $\mu \in \mathbb{Q}^3$.

All other implications are deduced false as they would otherwise contradict the last two cases by transitivity. □

Multi total-payoff is undecidable. In multi-dimension games, we have seen that the worst-case threshold problem for infimum mean-payoff is coNP-complete whereas it is in NP ∩ coNP for supremum mean-payoff. In both cases, $\mathcal{P}_1$ needs infinite memory to win, and memoryless strategies suffice for $\mathcal{P}_2$ [VCD+12].

The case of total-payoff objectives in multi-dimension games has never been considered before [CDRR13a, CDRR13b]. Surprisingly, the relation established

in Lemma 4.7 cannot be fully transposed in this context. We show that the threshold problem indeed becomes undecidable for multi-dimension games, even for a fixed number of dimensions.

Theorem 4.8. *The worst-case threshold problem for infimum and supremum total-payoff objectives is undecidable in multi-dimension games, for five dimensions.*

We reduce the halting problem for two-counter machines to the threshold problem for two-player total-payoff games with five dimensions. Counters take values $(v_1, v_2) \in \mathbb{N}^2$ along an execution, and can be incremented or decremented (if positive). A counter can be tested for equality to zero, and the machine can branch accordingly. We build a game with a sup. (resp. inf.) total-payoff objective of threshold $(0, 0, 0, 0, 0)$ for $\mathcal{P}_1$, in which $\mathcal{P}_1$ has to faithfully simulate an execution of the machine, and $\mathcal{P}_2$ can retaliate if he does not. We present gadgets by which $\mathcal{P}_2$ checks that (a) the counters are always non-negative, and that (b) a zero test is only passed if the value of the counter is really zero. The current value of counters (v_1, v_2) along an execution is encoded as the total sum of weights since the start of the game, $(v_1, -v_1, v_2, -v_2, -v_3)$, with v_3 being the number of steps of the computation. Hence, along a faithful execution, the 1st and 3rd dimensions are always non-negative, while the 2nd, 4th and 5th are always non-positive. To check that counters never go below zero, $\mathcal{P}_2$ is always able to go to an absorbing state with a self-loop of weight $(0, 1, 1, 1, 1)$ (resp. $(1, 1, 0, 1, 1)$). To check that all zero tests on counter 1 (resp. 2) are faithful, $\mathcal{P}_2$ can branch after a test to an absorbing state with a self-loop of weight $(1, 0, 1, 1, 1)$ (resp. $(1, 1, 1, 0, 1)$). Using these gadgets, $\mathcal{P}_2$ can punish an unfaithful simulation as he ensures that the sum in the dimension on which $\mathcal{P}_1$ has cheated always stays strictly negative and the outcome is thus losing (it is only the case if $\mathcal{P}_1$ cheats, otherwise all dimensions become non-negative). When an execution halts (with counters equal to zero w.l.o.g.) after a faithful execution, it goes to an absorbing state with weight $(0, 0, 0, 0, 1)$, ensuring a winning outcome for $\mathcal{P}_1$ for the total-payoff objective. If an execution does not halt, the 5th dimension stays strictly negative and the outcome is losing.

Proof. From a two-counter machine (2CM) $\mathcal{M}$, we construct a two-player game G

with five dimensions and an infimum (equivalently supremum) total-payoff objective such that $\mathcal{P}_1$ wins for threshold $(0,0,0,0,0)$ if and only if the 2CM halts.

A 2CM has two counters that can be incremented or decremented, and can test if their value is equal to zero (called zero test) and branch accordingly. The halting problem for 2CMs is undecidable [Min61]. Assume w.l.o.g. that we have a 2CM $\mathcal{M}$ such that if it halts, it halts with the two counters equal to zero.[3] In the game we construct, $\mathcal{P}_1$ has to faithfully simulate the 2CM $\mathcal{M}$. The role of $\mathcal{P}_2$ is to ensure that he does so by retaliating if it is not the case, hence making the outcome losing for the total-payoff objective.

The game is built as follows. The states of G are copies of the control states of $\mathcal{M}$ (plus some special states discussed in the following). Edges represent transitions between these states. The payoff function maps edges to 5-dimension vectors of the form $(c_1, -c_1, c_2, -c_2, d)$, that is, two dimensions for the first counter C_1, two for the second counter C_2, and one additional dimension. Each increment of counter C_1 (resp. C_2) in $\mathcal{M}$ is implemented in G as a transition of weight $(1,-1,0,0,-1)$ (resp. $(0,0,1,-1,-1)$. For decrements, we have weights respectively $(-1,1,0,0,-1)$ and $(0,0,-1,1,-1)$ for C_1 and C_2. Therefore, the current value of counters (v_1, v_2) along an execution of the 2CM $\mathcal{M}$ is represented in the game as the current sum of weights, $(v_1, -v_1, v_2, -v_2, -v_3)$, with v_3 the number of steps of the computation. The two dimensions per counter are used to enforce faithful simulation of non-negativeness of counters and zero test. The last dimension is decreased by one for every transition, except when the machine halts, from when it is incremented forever (i.e., the play in G goes to an absorbing state with self-loop $(0,0,0,0,1)$). This is used to ensure that a play in G is winning iff $\mathcal{M}$ halts.

We now discuss how this game G ensures faithful simulation of the 2CM $\mathcal{M}$ by $\mathcal{P}_1$.

- *Increment and decrement* of counter values are easily simulated using the first four dimensions.
- *Values of counters may never go below zero.* To ensure this, we allow $\mathcal{P}_2$ to branch after every step of the 2CM simulation to two special states, $s^1_{stop\_neg}$ and $s^2_{stop\_neg}$, which are absorbing and with self-loops of respective weights

[3] This is w.l.o.g. as it suffices to plug a machine that decreases both counters to zero at the end of the execution of the considered machine.

$(0,1,1,1,1)$ and $(1,1,0,1,1)$. If a negative value is reached on counter C_1 (resp. C_2), $\mathcal{P}_2$ can clearly win the game by branching to state $s^1_{stop\_neg}$ (resp. $s^2_{stop\_neg}$), as the total-payoff in the dimension corresponding to the negative counter will always stay strictly negative. On the contrary, if $\mathcal{P}_2$ decides to go to $s^1_{stop\_neg}$ (resp. $s^2_{stop\_neg}$) when the value of C_1 (resp. C_2) is positive, then $\mathcal{P}_1$ wins the game as this dimension will be positive and the other four will grow boundlessly. So these transitions are only used if $\mathcal{P}_1$ cheats.

- *Zero tests are correctly executed.* In the same spirit, we allow $\mathcal{P}_2$ to branch to two absorbing special states after a zero test, $s^1_{pos\_zero}$ and $s^2_{pos\_zero}$ with self-loops of weights $(1,0,1,1,1)$ and $(1,1,1,0,1)$. Such states are used by $\mathcal{P}_2$ if $\mathcal{P}_1$ cheats on a zero test (i.e., pass the test with a strictly positive counter value). Indeed, if a zero test was passed with the value of counter C_1 (resp. C_2) strictly greater than zero, then the current sum $(v_1, -v_1, v_2, -v_2, v_3)$ is such that $-v_1$ (resp. $-v_2$) is strictly negative. By going to $s^1_{pos\_zero}$ (resp. $s^2_{pos\_zero}$), $\mathcal{P}_2$ ensures that this sum will remain strictly negative in the considered dimension forever and the play is lost for $\mathcal{P}_1$.

Therefore, if $\mathcal{P}_1$ does not faithfully simulate $\mathcal{M}$, he is guaranteed to lose in G. On the other hand, if $\mathcal{P}_2$ stops a faithful simulation, $\mathcal{P}_1$ is guaranteed to win. It remains to argue that he wins iff the machine halts. Indeed, if the machine $\mathcal{M}$ halts, then $\mathcal{P}_1$ simulates its execution faithfully and either he is interrupted and wins, or the simulation ends in an absorbing state with a self-loop of weight $(0,0,0,0,1)$ and he also wins. Indeed, given that this state can only be reached with values of counters equal to zero (by hypothesis on the machine $\mathcal{M}$, without loss of generality), the running sum of weights will reach values $(0,0,0,0,n)$ where n grows to infinity, which ensures satisfaction of the infimum (and thus supremum) total-payoff objective for threshold $(0,0,0,0,0)$. On the opposite, if the 2CM $\mathcal{M}$ does not halt, $\mathcal{P}_1$ has no way to reach the halting state by means of a faithful simulation and the running sum in the fifth dimension always stays negative, thus inducing a losing play for $\mathcal{P}_1$, for both variants of the objective.

Consequently, we have that solving multi-dimension games for either the supremum or the infimum total-payoff objective is undecidable. □

Remark 4.9. Decidability of the worst-case threshold problem in total-payoff games for two, three or four dimensions is open. ◁

We end this section by noting that in multi-dimension total-payoff games, $\mathcal{P}_1$ may need infinite memory to win, even when all states belong to him ($S_2 = \emptyset$). Consider the game depicted in Fig. 4.4. As discussed in the proof of Lemma 4.7, given any threshold vector $\mu \in \mathbb{Q}^2$, $\mathcal{P}_1$ has a strategy to win the supremum total-payoff objective: it suffices to alternate between the two loops for longer and longer periods, each time waiting to get back above the threshold in the considered dimension before switching. This strategy needs infinite memory and actually, there exists no finite-memory strategy that can achieve a finite threshold vector: the negative amount to compensate grows boundlessly with each alternation, and thus no amount of finite memory can ensure to go above the threshold infinitely often.

4.3 Pure Finite-Memory Strategies

In general, multi mean-payoff requires infinite memory for $\mathcal{P}_1$, whereas for energy, finite memory is enough. With regard to practical applicability, we need strategies of $\mathcal{P}_1$ that can be *effectively implemented* in controllers. It is thus reasonable to retrict $\mathcal{P}_1$ to finite-memory strategies in mean-payoff games and look at the corresponding decision problem. This restriction does not change the situation of $\mathcal{P}_2$, for which memoryless strategies still suffice.

Considering games under pure finite-memory strategies (for both players) has the advantage of yielding *ultimately periodic plays*. Such plays share nice properties with regard to the mean-payoff objective: supremum and infimum variants coincide as the limit exists over them. With this restriction, we avoid unrealistic behaviors as witnessed in Example 4.5 under infinite-memory strategies.

Unfortunately, restricting $\mathcal{P}_1$ to finite-memory strategies is not sufficient in the total-payoff setting. The worst-case threshold problem does stay undecidable.

4.3.1 Mean-Payoff and Energy

In multi energy games, finite memory is already sufficient for $\mathcal{P}_1$ (Lemma 4.1), hence the unknown initial credit problem remains coNP-complete.

In multi mean-payoff games, restriction to finite memory permits to get back the equivalence with energy games that is well-known in the one-dimension context (see Sect. 2.3.2).

Lemma 4.10 ([VCD+12, Lemma 6]). *For all multi-dimension games, the answer to the unknown initial credit problem is* YES *if and only if the answer to the worst-case threshold problem*[4] *under finite-memory strategies is* YES.

The crux of this equivalence is that finite-memory winning strategies, both for energy and for mean-payoff objectives, induce non-negative (not necessarily simple) cycles. Such cycles are non-decreasing with regard to the energy level, and consequently, yield a non-negative mean-payoff. If such cycles cannot be guaranteed by a finite-memory strategy, then this strategy is losing for both objectives, as it means that strictly negative cycles on some dimensions can be forced by $\mathcal{P}_2$ (consuming an infinite amount of energy and inducing a strictly negative mean-payoff).

4.3.2 Total-Payoff

Our proof of undecidability (Thm. 4.8) relies on a reduction from the halting problem for two-counter machines. This proof extends easily when $\mathcal{P}_1$ is restricted to finite-memory strategies. Indeed, in the game based on the two-counter machine, if $\mathcal{P}_1$ wins, he wins with a finite-memory strategy that basically consists in implementing the machine. Clearly, if such a machine stops after a finite time, then it only requires a finite amount of memory for its computations.

Due to the undecidability of the total-payoff objectives in multi-dimension, we only focus on mean-payoff and energy games in Chap. 5 to 7. Nonetheless, we come back to the total-payoff objective in Part III, where we introduce alternative quantitative objectives, called *window objectives*, with similar flavor but preserving decidability in the multi-dimension setting.

4.3.3 Synthesis Complexity

From decision to synthesis. All the results presented before are for decision problems: the worst-case threshold problem for mean-payoff and the unknown

[4] Whenever the threshold is not specified, it should be understood as threshold $\{0\}^k$.

	MP	EG	par.	MP+par.	EG+par.
complexity	NP ∩ coNP				
$\mathcal{P}_1$ mem.	pure memoryless			pure infinite	pure pseudo-poly.
$\mathcal{P}_2$ mem.	pure memoryless				

Table 4.2: Overview of results on one-dimension quantitative objectives combined with parity.

initial credit problem for energy games. In practice, it is often needed not only to decide if $\mathcal{P}_1$ has a winning strategy, but also to actually *synthesize* such a winning strategy (see Sect. 2.2.3). This is a reason why restricting $\mathcal{P}_1$ to finite-memory strategies is important.

In the remaining of Part II (Chap. 5 to 7), we study the *synthesis problem* for finite-memory strategies in multi-dimension games with mean-payoff and energy objectives. We provide tight exponential bounds on memory and establish an optimal synthesis algorithm. These results are issued from joint work with Chatterjee and Raskin [CRR12a, CRR12b, CRR13].

Conjunction with parity. We present the first results considering the conjunction of multi-dimension quantitative objectives with parity objectives. All the results we provide extend to this more general setting. Observe that since we consider the synthesis of finite-memory strategies, it follows from Lemma 4.10 that multi-dimension energy with parity and multi-dimension mean-payoff with parity are equivalent.

Conjunction between parity and one-dimension quantitative objectives has been studied in the literature. We give an overview in Table 4.2. Mean-payoff parity games were studied in [CHJ05]: an exponential algorithm was given to decide if there exists a winning strategy for the worst-case threshold problem (which in general was shown to require infinite memory); and an improved algorithm was presented in [BMOU11]. Energy parity games were studied in [CD12]: it was shown that the unknown initial credit problem is in NP∩coNP, and an exponential algorithm was given. It was also shown that, for one-dimension energy parity objectives, finite-memory strategies with exponential memory (precisely, pseudo-polynomial) are sufficient, and that the decision problem for mean-payoff parity objective can be reduced to the one for energy parity objective. We use

some of these results in the following chapters, particularly in Chap. 7, where we show that randomness can replace memory in one-dimension mean-payoff parity games.

CHAPTER 5

Memory in Multi Energy Parity Games

Single Exponential Upper Bound ◇ Single Exponential Lower Bound ◇ Wrap-up

We establish *optimal memory bounds* for pure finite-memory winning strategies in multi energy parity games (MEPGs). As a corollary, we obtain results for pure finite-memory winning strategies on multi mean-payoff parity games (MMPPGs).

We first prove that single exponential memory is sufficient for winning strategies. Our result is a significant improvement over the triple exponential upper bound that can be obtained naively from the results known in literature [BJK10], even in the case of multi-dimension energy games without parity.

Additionally, we present how the parity condition in a MEPG can be removed by adding additional energy dimensions.

Second, we give a matching lower bound by presenting a family of game graphs where exponential memory is necessary in multi-dimension energy games (without parity), even when all the transition weights belong to $\{-1, 0, +1\}$.

All these results were obtained through collaboration with Chatterjee and Raskin [CRR12a, CRR12b, CRR13].

5.1 Single Exponential Upper Bound

Multi energy parity games. A sample game is depicted in Fig. 5.1. The key point in the upper bound proof on memory is to understand that for $\mathcal{P}_1$ to win a multi energy parity game, he must be able to force cycles whose energy level is positive in all dimensions and whose minimal parity is even. As stated in the next lemma, finite-memory strategies are sufficient for multi energy parity games for both players.

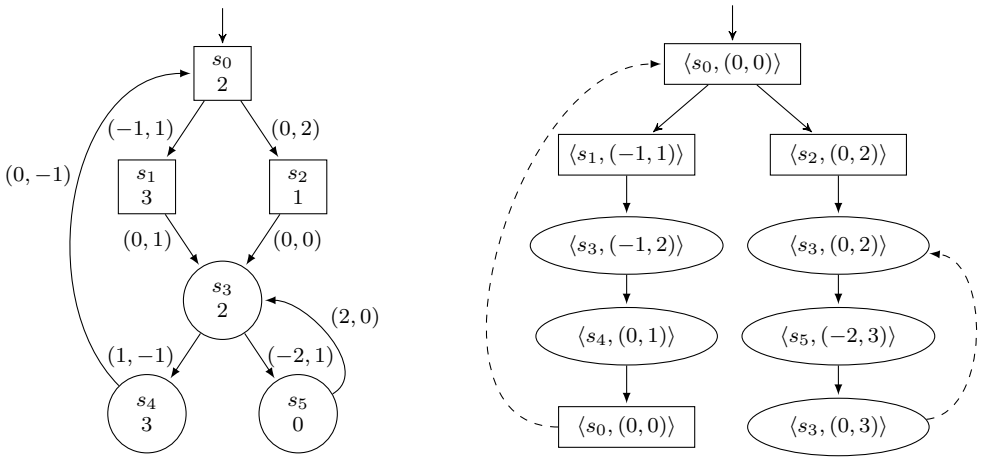


Figure 5.1: Two-dimension energy parity game and even-parity self-covering tree representing an arbitrary finite-memory winning strategy.

Lemma 5.1 (Extension of Lemma 4.1). *If $\mathcal{P}_1$ wins a multi energy parity game, then he has a pure finite-memory winning strategy. If $\mathcal{P}_2$ wins a multi energy parity game, then he has a pure memoryless winning strategy.*

Proof. The first part of the result follows using the standard well-quasi-ordering argument (straightforward extension of [CDHR10, Lemma 2]). The second part follows by the classical edge induction argument: [CDHR10, Lemma 3] and [CD12, Lemma 3] show the result using edge induction for multi energy and energy parity games, respectively. Repeating the arguments of [CD12, Lemma 3], and replacing the part on single energy objectives by the argument of [CDHR10, Lemma 3] for multi energy objectives, we obtain the desired result. □

By Lemma 5.1, we know that w.l.o.g. both players can be restricted to play pure finite-memory strategies. The property on the cycles can then be formalized as follows.

Lemma 5.2. *Let $G_p = (S_1, S_2, E, k, w, p)$ be a multi energy parity game and $s_{\mathsf{init}} \in S$ be an initial state. Let $\lambda_1^{pf} \in \Lambda_1^{PF}$ be a winning strategy of $\mathcal{P}_1$ for initial credit $v_0 \in \mathbb{N}^k$. Then, for all $\lambda_2^{pm} \in \Lambda_2^{PM}$, the outcome is a regular play $\pi = \rho \cdot (\eta_\infty)^\omega$, with $\rho \in \mathsf{Prefs}(G_p)$, $\eta_\infty \in S^+$, such that $\mathsf{EL}(\eta_\infty) \geq 0$ and $\mathsf{Par}(\pi) = \min\{p(s) \mid s \in \eta_\infty\}$ is even.*

Proof. Recall that both players play with pure finite memory strategies. Therefore, a finite number of decisions are made and the outcome is a regular (i.e., ultimately periodic) play $\pi = \rho \cdot (\eta_\infty)^\omega$. Note that $\mathsf{EL}(\rho)$ does not have to be positive, as $\mathcal{P}_1$ may have $v_0 > \mathsf{EL}(\rho)$. Similarly, priorities of states visited in ρ have no impact on winning as they are only visited a finite number of times. First, suppose $\mathsf{EL}(\eta_\infty) < 0$ on some dimension $1 \leq j \leq k$. Then, after $m > 0$ cycles, for some $n > 0$, the energy level will be $\mathsf{EL}(\pi(n)) = \mathsf{EL}(\rho \cdot (\eta_\infty)^m) = \mathsf{EL}(\rho) + m \cdot \mathsf{EL}(\eta_\infty)$. Since v_0 is finite and $m \to \infty$, there exist some $m, n > 0$, such that $v_0 + \mathsf{EL}(\pi(n)) < 0$ on dimension j and λ_1 is not winning. Second, suppose $\min\{p(s) \mid s \in \eta_\infty\}$ is odd. Since the set of states visited infinitely often is exactly the set of states in η_∞, this implies that $\mathsf{Par}(\pi)$ is odd, and thus λ_1 is not winning. □

A *self-covering path* in a game, straightforwardly extending the notion introduced by Rackoff [Rac78] for *vector addition systems* (VAS), is a sequence of states $s_0 s_1 s_2 \dots s_m$ such that there exist two positions i and j that verify $0 \leq i < j \leq m$, $s_i = s_j$ and $\mathsf{EL}(s_0 \dots s_i) \leq \mathsf{EL}(s_0 \dots s_i \dots s_j)$. In other words, such a path describes a finite prefix followed by a cycle which has a non-negative effect on the energy level. Ensuring such cycles is crucial to win the energy objective. With the notion of regular play of Lemma 5.2, we generalize the notion of self-covering path to include the parity condition. We show here that, if such a path exists, then the lengths of its cycle and the prefix needed to reach it can be bounded. Bounds on the strategy follow. In [Rac78], Rackoff showed how to bound the length of self-covering paths in VAS. This work was extended to vector addition systems with states (VASS) by Rosier and Yen [RY86]. Recently, Brázdil et al. introduced reachability games on VASS and the notion of

self-covering trees [BJK10]. Their Zero-safety problem with ω initial marking is equivalent to multi energy games with weights in $\{-1, 0, 1\}$, and without the parity condition. They showed that if winning strategies exist for $\mathcal{P}_1$, then some of them can be represented as *self-covering trees* of bounded depth. Trees have to be considered instead of paths, as in a game setting all the possible choices of the adversary ($\mathcal{P}_2$) must be considered. Here, we extend the notion of self-covering trees to *even-parity self-covering trees*, in order to handle parity objectives.

Definition 5.3 (Even-parity self-covering tree). An *even-parity self-covering tree* (epSCT) for $s \in S$ is a finite tree $T = (Q, R)$, where Q is the set of nodes, $\Theta \colon Q \to S \times \mathbb{Z}^k$ is a labeling function and $R \subset Q \times Q$ is the set of edges, such that

- The root of T is labeled $\langle s, (0, \dots, 0)\rangle$.
- If $\varsigma \in Q$ is not a leaf, then let $\Theta(\varsigma) = \langle t, u\rangle$, $t \in S$, $u \in \mathbb{Z}^k$, such that
  - if $t \in S_1$, then ς has a unique child ϑ such that $\Theta(\vartheta) = \langle t', u'\rangle$, $(t, t') \in E$ and $u' = u + w(t, t')$;
  - if $t \in S_2$, then there is a bijection between children of ς and edges of the game leaving t, such that for each successor $t' \in S$ of t in the game, there is one child ϑ of ς such that $\Theta(\vartheta) = \langle t', u'\rangle$, $u' = u + w(t, t')$.
- If ς is a leaf, then let $\Theta(\varsigma) = \langle t, u\rangle$ such that there is some ancestor ϑ of ς in T such that $\Theta(\vartheta) = \langle t, u'\rangle$, with $u' \leq u$, and the downward path from ϑ to ς, denoted by $\vartheta \rightsquigarrow \varsigma$, has minimal priority even. We say that ϑ is an *even-descendance energy ancestor* of ς.

Intuitively, each path from root to leaf is a self-covering path of even parity in the game graph so that plays unfolding according to such a tree correspond to winning plays of Lemma 5.2. Thus, the epSCT fixes how $\mathcal{P}_1$ should react to actions of $\mathcal{P}_2$ in order to win the MEPG (Fig. 5.1). Note that as the tree is finite, one can take the largest negative number that appears on a node in each dimension to compute an initial credit for which there is a winning strategy (i.e., the one described by the tree). In particular, recall W denote the maximal absolute weight appearing on an edge in G_p. Then, for an epSCT T of depth l, it is straightforward to see that the maximal initial credit required is at most $l \cdot W$

as the maximal decrease at each level of the tree is bounded by W. We suppose $W > 0$ as otherwise, any strategy of $\mathcal{P}_1$ is winning for the energy objective, for any initial credit vector $v_0 \in \mathbb{N}^k$.

Let us explicitely state how $\mathcal{P}_1$ can deploy a strategy $\lambda_1^T \in \Lambda_1^{PF}$ based on an epSCT $T = (Q, R)$. We refer to such a strategy as an *epSCT strategy*. It consists in following a path in the tree T, moving a pebble from node to node and playing in the game depending on edges taken by this pebble. Each time a node ς such that $\Theta(\varsigma) = \langle t, u \rangle$ is encountered, we do the following.

- If ς is a leaf, the pebble directly goes up to its oldest even-descendance energy ancestor ϑ. By oldest we mean the first encountered when going down in the tree from the root. Note that this choice is arbitrary, in an effort to ease following proof formulations, as any one would suit.

- Otherwise, if ς is not a leaf,
  - if $t \in S_2$ and $\mathcal{P}_2$ plays state $t' \in S$, the pebble is moved along the edge going to the only child ϑ of ς such that $\Theta(\vartheta) = \langle t', u' \rangle$, $u' = u + w(t, t')$;
  - if $t \in S_1$, the pebble moves to ϑ, $\Theta(\vartheta) = \langle t', u' \rangle$, the only child of ς, and $\mathcal{P}_1$ strategy is to choose the state t' in the game.

If such an epSCT T of depth l exists for a game G_p, then $\mathcal{P}_1$ can play the strategy $\lambda_1^T \in \Lambda_1^{PF}$ to win the game with initial credit bounded by $l \cdot W$.

Bounding the depth of epSCTs. Consider a multi energy game *without* parity. Then, the priority condition on downward paths from ancestor to leaf is not needed and self-covering trees (i.e., epSCTs without the condition on priorities) suffice to describe winning strategies. One can bound the size of SCTs using results on the size of solutions for linear diophantine equations (i.e., with integer variables) [BT76]. In particular, recent work on reachability games over VASS with weights $\{-1, 0, 1\}$ [BJK10, Lemma 7] states that if $\mathcal{P}_1$ has a winning strategy on a VASS, then he can exhibit one that can be described as an SCT whose *depth* is at most $l = 2^{(d-1)\cdot|S|} \cdot (|S| + 1)^{c \cdot k^2}$, where c is a constant independent of the considered VASS and d its branching degree (i.e., the highest number of outgoing edges on any state).

Naive use of this bound for multi energy games with arbitrary integer weights would induce a *triple* exponential bound for memory. A straightforward translation of a game with arbitrary weights into an equivalent game that uses only weights in $\{-1, 0, 1\}$ induces a blow-up by W in the size of the state space, and thus an exponential blow-up by W in the depth of the tree, which becomes doubly exponential as we have

$$l = 2^{(d-1)\cdot W\cdot|S|} \cdot (W \cdot |S| + 1)^{c\cdot k^2} = 2^{(d-1)\cdot 2^V\cdot|S|} \cdot (W \cdot |S| + 1)^{c\cdot k^2},$$

where V denotes the number of bits used by the encoding of W. Moreover, the width of the tree increases as d^l, i.e., it increases exponentially with the depth. So straight application of previous results provides an overall tree of triple exponential size. In this chapter, we improve this bound and prove a single exponential upper bound, even for multi energy *parity* games. We proceed in two steps, first studying the depth of the epSCT, and then showing how to compress the tree into a *directed acyclic graph* (DAG) of *single* exponential size.

Lemma 5.4. *Let $G_p = (S_1, S_2, E, k, w, p)$ be a multi energy parity game such that W is the maximal absolute weight appearing on an edge and d the branching degree of G_p. Let $s_{\mathsf{init}} \in S$ be an initial state. Suppose there exists a finite-memory winning strategy for $\mathcal{P}_1$. Then there is an even-parity self-covering tree for s_{init} of depth at most $l = 2^{(d-1)\cdot|S|} \cdot (W \cdot |S| + 1)^{c\cdot k^2}$, where c is a constant independent of G_p.*

Lemma 5.4 eliminates the exponential blow-up in depth induced by a naive coding of arbitrary weights into $\{-1, 0, 1\}$ weights, and implies an overall doubly exponential upper bound. Our proof is a generalization of [BJK10, Lemma 7], using a more refined analysis to handle both *parity* and *arbitrary integer weights*. The idea is as follows. First, consider the one-player case. The epSCT is reduced to a path. By Lemma 5.2, it is composed of a finite prefix, followed by an infinitely repeated sequence of positive energy level and even minimal priority. We bound the length of this sequence by eliminating cycles that are not needed for energy or parity. Second, to extend the result to two-player games, we use an induction on the number of choices available for $\mathcal{P}_2$ in a given state. Intuitively, we show that if $\mathcal{P}_1$ can win with an epSCT T_A when $\mathcal{P}_2$ plays edges from a set A in a state s, and if he can also win with an epSCT T_B when $\mathcal{P}_2$ plays edges from

a set B, then he can win when $\mathcal{P}_2$ chooses edges from both A and B, with an epSCT whose depth is bounded by the sum of depths of T_A and T_B.

Proof. The proof is made in two steps. First, we consider the one-player case, where $S_2 = \emptyset$. Second, we use an induction scheme over the choice degree of $\mathcal{P}_2$ to extend our results to the two-player case.

We start with $S_2 = \emptyset$, the one-player game. By Lemma 5.2, a winning play is of the form $\pi = \rho \cdot (\eta_\infty)^\omega$ such that $\mathsf{EL}(\eta_\infty) \geq 0$ and $\mathsf{Par}(\pi) = \min\{p(s) \mid s \in \eta_\infty\}$ is even. Notice that such a play corresponds to the epSCT defined above, as it reduces to an even-parity self-covering path $\langle s_{init}, (0, \ldots, 0)\rangle \rightsquigarrow \langle s, u\rangle \rightsquigarrow \langle s, u'\rangle$ with $u' \geq u$. Therefore its existence is guaranteed and it remains to bound its length. Given such a path, the idea is to eliminate unnecessary cycles, in order to reduce its length while maintaining the needed properties (i.e., positive energy and even minimal priority). First, notice that cycles in the sub-path $\langle s_{init}, (0, \ldots, 0)\rangle \rightsquigarrow \langle s, u\rangle$ can be trivially erased as they are only visited a finite number of times and thus (a) the initial credit can compensate for the loss of their potential positive energy effect, and (b) they do not contribute in the parity. Now consider the sub-path $\langle s, u\rangle \rightsquigarrow \langle s, u'\rangle$. Since it induces a winning play, its minimal priority is even. Let p_m be this priority. We may suppose w.l.o.g. that $p(s) = p_m$, otherwise it suffices to shift this sub-path to $\langle s', v\rangle \rightsquigarrow \langle s', v'\rangle$ for some state s' such that $p(s') = p_m$ and $v' \geq v$, and add the sub-path $\langle s, u\rangle \rightsquigarrow \langle s', v\rangle$ to the finite prefix. Now we may eliminate each cycle of $\langle s, u\rangle \rightsquigarrow \langle s, u'\rangle$ safely in regards to the parity objective as they only contain states with greater or equal priority. Thus, we only need to take care of the energy, and fall under the scope of [BJK10, Lemma 15] for the special case of weights in $\{-1, 0, 1\}$, where an upper bound $h(|S|, k) = (|S| + 1)^{c \cdot k^2}$ on the length of such a path is shown.

For a one-player game G with weights in $\{-W, -W+1, \ldots, W-1, W\}$, we claim that an upper bound $h(W, |S|, k) = (W \cdot |S| + 1)^{c \cdot k^2}$ is obtained. Indeed, one can translate $G_p = (S_1, S_2, E, k, w, p)$ into an equivalent game $G'_{p'} = (S'_1, S_2, E', k, w', p')$ such that each edge of G_p is split into at most W edges in $G'_{p'}$, with at most $(W-1)$ dummy states in between, so that each edge of $G'_{p'}$ only uses weights in $\{-1, 0, 1\}$. Let S_d denote the set of these added dummy

states. We define this translation $\mathsf{Tr}\colon G_p \mapsto G'_{p'}$ as follows:

$$\mathsf{Tr}(S_1) = S_1 \cup S_d, \qquad \mathsf{Tr}(S_2) = S_2, \qquad \mathsf{Tr}(s_{\mathsf{init}}) = s_{\mathsf{init}},$$
$$\mathsf{Tr}(E) = \bigcup_{(s,t)\in E} \mathsf{Tr}((s,t)), \qquad \mathsf{Tr}(k) = k, \qquad \mathsf{Tr}(w) = w'\colon E' \to \{-1,0,1\}^k.$$

Finally, we define $\mathsf{Tr}(p) = p'\colon S' \to \mathbb{N}$ such that for all $(s,t) \in E$ such that $m = \max\{w(s,t)(j) \mid 1 \leq j \leq k\} - 1$, we have that

$$\mathsf{Tr}\left((s,t)\right) = \left\{(s, s_d^1), (s_d^1, s_d^2), \ldots, (s_d^{m-1}, s_d^m), (s_d^m, t)\right\}$$

such that

$$\Big(\forall\, j > 0,\ s_d^j \in S_d \ \wedge\ p'(s_d^j) = p(s)\Big) \ \wedge \sum_{(q,r)\in \mathsf{Tr}((s,t))} w'(q,r) = w(s,t).$$

To be formally correct, we have to add that for all $s_d \in S_d$, we have $\mathsf{degree}_{in}(s_d) = \mathsf{degree}_{out}(s_d) = 1$, and for all $s \notin S_d$, we have $p'(s) = p(s)$. This translation does not hinder the outcome of the game as each edge in G_p has a unique corresponding path in $G'_{p'}$ that preserves the weights and the visited priorities, and that offers no added choice to $\mathcal{P}_1$. Since G_p possesses $|E| \leq |S|^2$ edges, and for each edge of G_p, we add at most $(W-1)$ dummy states in $G'_{p'}$, we have $|S'| \leq |S| + |S|^2 \cdot (W-1) \leq |S|^2 \cdot W$. Therefore, by applying [BJK10, Lemma 15] on $G'_{p'}$, we obtain the following upper bound:

$$h\left(W, |S|, k\right) = h\left(|S'|, k\right) = \left(|S|^2 \cdot W + 1\right)^{c \cdot k^2} = \left(W \cdot |S| + 1\right)^{c' \cdot k^2}$$

for some constant c' that is independent of G_p.

Now, consider $S_2 \neq \emptyset$. (i) We extend [BJK10, Lemma 16] for parity. This will help us to establish an induction scheme over the choice degree of $\mathcal{P}_2$. Suppose $s \in S_2$ has more than one outgoing edge. Let $\tau = (s,t) \in E$ be one of them and $R \subset E$ denote the nonempty set of other outgoing edges. Let G_p^τ (resp. G_p^R) be the game induced when removing R (resp. τ) from G_p. Suppose that (a) s is winning for $\mathcal{P}_1$ in G_p^R for initial credit $v_R \in \mathbb{N}^k$, and (b) there exists some state $s' \in S$ such that s' is winning for $\mathcal{P}_1$ in G_p^τ for initial credit $v_\tau \in \mathbb{N}^k$. We claim that s' is winning in G_p for initial credit $v_0 = v_\tau + v_R$. Indeed, let λ_1^τ and λ_1^R

resp. denote winning strategies for $\mathcal{P}_1$ in G_p^τ and G_p^R. Let $\mathcal{P}_1$ use the following strategy. Player $\mathcal{P}_1$ plays λ_1^τ as long as $\mathcal{P}_2$ does not play any edge of R. If such an edge is played, then $\mathcal{P}_1$ switches to strategy λ_1^R and plays it until edge τ is played again by $\mathcal{P}_2$, in which case $\mathcal{P}_1$ switches back to λ_1^τ, and so on. In this way, the outcome of the game is guaranteed to be a play $\pi = s' \dots s \dots s \dots s \dots$ resulting from a merge between a play consistent with λ_1^τ over G_p^τ (whose energy level is bounded by $-v_\tau$ at all times), and a play consistent with λ_1^R over G_p^R (whose energy level is bounded by $-v_R$ at all times). Therefore, the combined overall energy level of any prefix ρ of this play is bounded by $(-v_\tau - v_R)$ as positive cycles in G_p^τ and G_p^R do remain positive in G_p. Furthermore, the parity condition is preserved in G_p. Indeed, suppose it is not. Thus, there exists a state visited infinitely often in the outcome such that its priority is minimal and odd. However, as the outcome results from merging plays resp. consistent with λ_1^τ and λ_1^R, this implies that one of those strategies yields an odd minimal priority, which contradicts the fact that they are winning. This proves the claim.

(ii) We apply the induction scheme of [BJK10, Lemma 18] on $r = |\{(s,t) \in E \mid s \in S_2\}| - |S_2| \leq (d-1) \cdot |S|$, the choice degree of $\mathcal{P}_2$. Notice that our translation $\mathsf{Tr}\colon G_p \mapsto G'_{p'}$ maintains this choice degree unchanged. The claim is that for a winning state s', there is an epSCT of depth bounded by $2^r \cdot h(W, |S|, k)$. We have proved that for the base case $r = 0$, similar to $S_2 = \emptyset$, this claim is true. So assume it holds for r, it remains to prove that it is preserved for $r+1$. Let $s \in S_2$ be such that $\mathcal{P}_2$ has at least two outgoing edges. As before, we define G_p^τ and G_p^R. Clearly, the choice degree of $\mathcal{P}_2$ is at most r in both games. Let s' be a winning state in G_p. As $\mathcal{P}_2$ has less choices in both G_p^τ and G_p^R, clearly s' is still winning in those games. If an epSCT in either of them (which are guaranteed to exist and have depth bounded by $2^r \cdot h(W, |S|, k)$ by hypothesis) do not contain the state s, then the claim is verified. Now suppose we have two epSCTs for games G_p^τ and G_p^R such that they both contain state s. Notice that s is winning in those two games and as such, is the root of two respective epSCTs of depth less than $2^r \cdot h(W, |S|, k)$. Applying (i) on states s' and s, we get an epSCT for s' in G_p of depth $2 \cdot 2^r \cdot h(W, |S|, k)$, which concludes the proof. □

From multi energy parity games to multi energy games. Let G_p be a MEPG and assume that $\mathcal{P}_1$ has a winning strategy in that game. By Lemma 5.4,

there exists an epSCT whose depth is bounded by l. As a direct consequence of that bounded depth, we have that $\mathcal{P}_1$, by playing the strategy prescribed by the epSCT, enforces a stronger objective than the parity objective. Namely, this strategy ensures to "never visit more than l states of odd priorities before seeing a smaller even priority" (which is a safety objective). Then, the parity condition can be transformed into additional energy dimensions.

While our transformation shares ideas with the classical transformation of parity objectives into safety objectives, first proposed in [BJW02] (look also at [DR11, Lemma 6.4]), it is technically different because energy levels cannot be reset (as it would be required by those classical constructions). The reduction is as follows. For each odd priority, we add one dimension. The energy level in this dimension is decreased by 1 each time this odd priority is visited, and it is increased by l each time a smaller even priority is visited. If $\mathcal{P}_1$ is able to maintain the energy level positive for all dimensions (for a given initial energy level), then he is clearly winning the original parity objective; on the other hand, an epSCT strategy that wins the original objective also wins the new game.

Lemma 5.5. *Let $G_p = (S_1, S_2, E, k, w, p)$ be a multi energy parity game with priorities in $\{0, 1, \ldots, 2 \cdot m\}$, such that W is the maximal absolute weight appearing on an edge. Then we can construct a multi energy game G with the same set of states, $(k + m)$ dimensions and a maximal absolute weight bounded by l, as defined by Lemma 5.4, such that $\mathcal{P}_1$ has a winning strategy in G iff he has one in G_p.*

Proof. Let $G_p = (S_1, S_2, E, k, w, p)$ be a MEPG with priorities in $\{0, 1, \ldots, 2 \cdot m\}$. Let $G = (S_1, S_2, E, (k + m), w')$ be the multi energy game (MEG) obtained by applying the following transformation: $\forall\ (s, t) \in E$, $\forall\ 1 \leq j \leq k$, $w'((s, t))(j) = w((s, t))(j)$, and (a) if $p(t)$ is even, $\forall\ k < j \leq \frac{p(t)}{2}$, $w'((s, t))(j) = 0$ and $\forall\ p(t) < j \leq k + m$, $w'((s, t))(j) = l$, or (b) if $p(t)$ is odd, $\forall\ k < j \leq k + m$, $j \neq \frac{p(t)}{2}$, $w'((s, t))(j) = 0$ and $w'((s, t))(\frac{p(t)}{2}) = -1$. We have to prove both ways of the equivalence.

First, suppose $\lambda_1 \in \Lambda_1^{PF}$ is a winning strategy for $\mathcal{P}_1$ in the MEPG G_p. By Lemma 5.4, there is an epSCT of depth at most l for initial state $s_{\mathsf{init}} \in S$. Thus, we know that in every repeated sequence of l states, the minimal visited priority will be even. Therefore, for all additional dimensions, ranging from $k+1$ to $k+m$, the effect of a sequence of l states will be bounded from below by $-1 \cdot (l - 1) + l$,

which is positive. Thus strategy λ_1 is also winning in G (with initial credit bounded by l on additional dimensions).

Second, suppose $\lambda_1 \in \Lambda_1^{PF}$ is a winning strategy for $\mathcal{P}_1$ in the MEG G, as defined above. Since λ_1 is winning, it yields an SCT (epSCT without the parity condition) of bounded depth such that $\mathcal{P}_1$ is able to enforce positive energy cycles. By definition of weights over G, this cannot be the case if the minimal priority infinitely often visited is odd. Thus this strategy is winning for parity on G_p, and stays winning for energy over dimensions 1 to k as weights are unchanged. □

Bounding the width. Thanks to Lemma 5.5, we continue with multi energy games without parity. In order to bound the overall size of memory for winning strategies, we consider the width of self-covering trees. The following lemma states that SCTs, whose width is at most doubly exponential by application of Lemma 5.4, can be compressed into *directed acyclic graphs* (DAGs) of single exponential width. Thus we eliminate the second exponential blow-up and give an overall single exponential bound for memory of winning strategies.

Lemma 5.6. *Let $G = (S_1, S_2, E, k, w)$ be a multi energy game such that W is the maximal absolute weight appearing on an edge and d the branching degree of G. Suppose there exists a finite-memory winning strategy for $\mathcal{P}_1$. Then, there exists $\lambda_1^D \in \Lambda_1^{PF}$ a winning strategy for $\mathcal{P}_1$ described by a DAG D of depth at most $l = 2^{(d-1)\cdot|S|} \cdot (W \cdot |S| + 1)^{c \cdot k^2}$ and width at most $L = |S| \cdot (2 \cdot l \cdot W + 1)^k$, where c is a constant independent of G. Thus the overall memory needed to win this game is bounded by the single exponential $l \cdot L$.*

The sketch of this proof is the following. By Lemma 5.4, we know that there exists a tree T, and thus a DAG, that satisfies the bound on depth. We construct a finite sequence of DAGs, whose first element is T, so that (1) each DAG describes a winning strategy for the same initial credit, (2) each DAG has the same depth, and (3) the last DAG of the sequence has its width bounded by $|S| \cdot (2 \cdot l \cdot W + 1)^k$. This sequence $D_0 = T, D_1, D_2, \ldots, D_n$ is built by merging nodes on the same level of the initial tree depending on their labels, level by level. The key idea of this procedure is that what actually matters for $\mathcal{P}_1$ is only the current energy level, which is encoded in node labels in the self-covering tree T. Therefore, we merge nodes with identical states and energy levels: since $\mathcal{P}_1$

can essentially play the same strategy in both nodes, we only keep one of their subtrees.

It is possible to further reduce the practical size of the compressed resulting DAG by merging nodes according to a "greater or equal" relation over energy levels rather than simply equality (Fig. 5.2). This improvement is part of the algorithm that follows, and it has a significant impact on the practical width of DAGs as it can then be bounded by the number of *incomparable* labeling vectors instead of *unequivalent* ones.

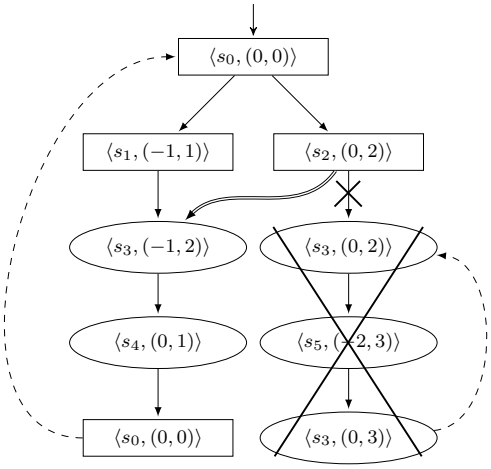


Figure 5.2: Merge between comparable nodes.

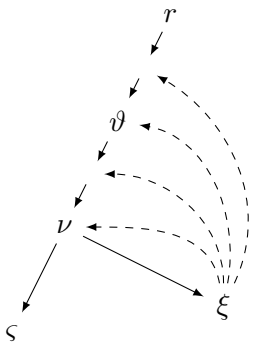


Figure 5.3: Cycles have positive energy levels.

The remainder of this section is dedicated to the proof of Lemma 5.6. We need to introduce some notations and two intermediate lemmas. If he so wishes, the reader may directly proceed to the Sect. 5.2 and Lemma 5.9 for results on lower memory bounds.

We first introduce some notations. Let $T = (Q, R)$ be a self-covering tree (i.e., epSCT without the parity condition). We define the partial order $\preceq$ on Q such that for all $\varsigma_1, \varsigma_2 \in Q$ such that $\Theta(\varsigma_1) = \langle t_1, u_1\rangle$ and $\Theta(\varsigma_2) = \langle t_2, u_2\rangle$, we have $\varsigma_1 \preceq \varsigma_2$ iff $t_1 = t_2$ and $u_1 \leq u_2$. We denote the equivalence by $\simeq$ such that $\varsigma_1 \simeq \varsigma_2$ iff $\varsigma_1 \preceq \varsigma_2$ and $\varsigma_2 \preceq \varsigma_1$. For all $\varsigma \in Q$, let Anc and EnAnc resp. denote the set of *ancestors* and *energy ancestors* of ς in T: $\mathsf{Anc}(\varsigma) = \{\vartheta \in Q \setminus \{\varsigma\} \mid \vartheta \vDash \exists\Diamond\varsigma\}$, where we use the classical CTL notation [CE81] to denote that there exists a

path from ϑ to ς in T, and $\mathsf{EnAnc}(\varsigma) = \{\vartheta \in \mathsf{Anc}(\varsigma) \mid \vartheta \preceq \varsigma\}$.

We build a sequence of DAGs $(D_i)_{0 \leq i \leq n} \equiv D_0 = T, D_1, D_2, \ldots, D_n$ such that for all $0 < i \leq n$, D_i is obtained from D_{i-1} by *merging* two equivalent nodes of the same minimal level (i.e., closest to the root) of D_{i-1}. The sequence stops when we obtain a DAG $D_n = (Q_n, R_n)$ such that for all level j of D_n, there does not exist two distinct equivalent nodes on level j. This construction induces merges by increasing depth, starting with level one. Moreover, if a DAG D_i of the sequence is the result on merges up to level j, then it has the tree property (i.e., every node has a unique father) for levels greater than j. As the depth and the branching degree of T are finite, the defined sequence of DAGs is finite (and actually bounded).

Let us give a formal definition of the *merge* operation. Consider such a DAG $D_i = (Q_i, R_i)$. Let j the minimal level of D_i that contains two equivalent nodes. Let $\varsigma_1, \varsigma_2 \in Q_i(j)$ (i.e., nodes of level j) be two nodes such that $\varsigma_1 \neq \varsigma_2$ and $\varsigma_1 \simeq \varsigma_2$. We suppose w.l.o.g. an arbitrary order on nodes of the same level so that ς_1, ς_2 are the two leftmost nodes that satisfy this condition. We define $D_{i+1} = (Q_{i+1}, R_{i+1}) = \mathsf{merge}(D_i)$ as the result of the following transformation:

- $Q_{i+1} = Q_i \setminus (\{\varsigma_2\} \cup \{\varsigma_d \in Q_i \mid \varsigma_2 \in \mathsf{Anc}(\varsigma_d)\})$,
- $R_{i+1} = (R_i \cap (Q_{i+1} \times Q_{i+1})) \cup \{(\vartheta, \varsigma_1) \mid (\vartheta, \varsigma_2) \in R_i\}$.

Thus, we eliminate the subtree starting in ς_2 and replace all edges that point to ς_2 by edges pointing to ς_1. This follows the idea that the same strategy can be played in ς_2 as in ς_1 since the present state and the energy level are the same.

Let $D_i = (Q_i, R_i)$ be a DAG of the sequence $(D_i)_{0 \leq i \leq n}$. Given $\varsigma \in Q_i$, $\vartheta \in \mathsf{Anc}(\varsigma)$, we denote by $\vartheta \rightsquigarrow \varsigma$ an arbitrary downward path from ϑ to ς in D_i. Given a leaf $\varsigma \in Q_i$, we denote its oldest energy ancestor by $\mathsf{oea}(\varsigma)$. Recall that a strategy is described by such a DAG according to moves of a pebble. Given a leaf $\varsigma \in Q_i$ and one of its energy ancestors $\vartheta \in \mathsf{EnAnc}(\varsigma)$, we represent the pebble going up from ς to ϑ by $\varsigma \circlearrowleft \vartheta$. Given $\alpha, \beta \in (Q_i)^*$, $\alpha \circlearrowleft \beta$ naturally extends this notation such that we have $\mathsf{Last}(\alpha) \circlearrowleft \mathsf{First}(\beta)$. We consider energy levels of paths in the tree by refering to their counterparts in the game. Note that given $\vartheta, \varsigma \in Q_i$, $\Theta(\vartheta) = \langle t, u \rangle$, $\Theta(\varsigma) = \langle t', u' \rangle$, we have $\mathsf{EL}(\vartheta \rightsquigarrow \varsigma) = u' - u$. We start with two useful lemmas.

Lemma 5.7. *Let $D_i = (Q_i, R_i)$ be a DAG of $(D_i)_{0\leq i\leq n}$. For all nodes $\varsigma_1, \varsigma_2 \in Q_i$ such that $\varsigma_1 \simeq \varsigma_2$, we have that $\forall\, \vartheta \in \mathsf{Anc}(\varsigma_1) \cap \mathsf{Anc}(\varsigma_2)$, $\mathsf{EL}(\vartheta \rightsquigarrow \varsigma_1) = \mathsf{EL}(\vartheta \rightsquigarrow \varsigma_2)$.*

Proof. The proof is straightforward. □

Lemma 5.8. *Let $D_i = (Q_i, R_i)$ be a DAG of $(D_i)_{0\leq i\leq n}$. Let $\varsigma, \vartheta, \nu, \xi \in Q_i$ be four nodes such that ς and ξ are leafs, ν is the deepest common ancestor of ς and ξ, and ϑ is an ancestor of ν. Let the oldest energy ancestor of ξ be an ancestor of ς, i.e., $\mathsf{oea}(\xi) \in \mathsf{Anc}(\varsigma)$. We have that $\mathsf{EL}(\vartheta \rightsquigarrow \varsigma) \leq \mathsf{EL}(\vartheta \rightsquigarrow \nu \rightsquigarrow \xi \circlearrowleft \mathsf{oea}(\xi) \rightsquigarrow \varsigma)$.*

This lemma states that we can extract pebble cycles, which have positive energy levels, from a given path, in order to obtain some canonical path whose energy level is lower or equal (Fig. 5.3).

Proof. Let $\chi = \mathsf{oea}(\xi)$ and $\rho = \vartheta \rightsquigarrow \nu \rightsquigarrow \xi \circlearrowleft \chi \rightsquigarrow \varsigma$. Since $\chi \in \mathsf{Anc}(\varsigma) \cap \mathsf{Anc}(\xi)$, we have $\chi \in \mathsf{Anc}(\nu) \cup \{\nu\}$. Therefore, and applying Lemma 5.7, four cases are possible: $\chi \in \mathsf{Anc}(\vartheta)$, $\chi = \vartheta$, $\chi \in \mathsf{Anc}(\nu) \setminus (\mathsf{Anc}(\vartheta) \cup \{\vartheta\})$, and $\chi = \nu$. Consider the first case, $\chi \in \mathsf{Anc}(\vartheta)$. Then $\rho = \vartheta \rightsquigarrow \nu \rightsquigarrow \xi \circlearrowleft \chi \rightsquigarrow \vartheta \rightsquigarrow \nu \rightsquigarrow \varsigma$. We have $\mathsf{EL}(\rho) = \mathsf{EL}(\vartheta \rightsquigarrow \nu) + \mathsf{EL}(\nu \rightsquigarrow \xi) + \mathsf{EL}(\chi \rightsquigarrow \vartheta) + \mathsf{EL}(\vartheta \rightsquigarrow \nu) + \mathsf{EL}(\nu \rightsquigarrow \varsigma) = \mathsf{EL}(\chi \rightsquigarrow \vartheta \rightsquigarrow \nu \rightsquigarrow \xi) + \mathsf{EL}(\vartheta \rightsquigarrow \varsigma)$. By definition of $\chi = \mathsf{oea}(\xi)$, the first term is positive. Thus, $\mathsf{EL}(\rho) \geq \mathsf{EL}(\vartheta \rightsquigarrow \varsigma)$. Arguments are similar for the other cases. □

We proceed with the proof of Lemma 5.6.

Proof of Lemma 5.6. Let $(D_i)_{0\leq i\leq n}$ be the sequence of DAGs defined above. We claim that (i) each DAG describes a winning strategy for the same initial credit, (ii) each DAG has the same depth l, and (iii) the last DAG of the sequence has its width bounded by $|S| \cdot (2 \cdot l \cdot W + 1)^k$.

(i) First, recall that $\mathcal{P}_1$ can play a strategy $\lambda_1^T \in \Lambda_1^{PF}$ based on edges taken by a pebble on T. Notice that moving the pebble as we previously defined is possible because nodes belonging to $\mathcal{P}_1$ have only one child, and nodes of $\mathcal{P}_2$ have childs covering all his choices once, and only once. Fortunately, the merge operation maintains this property. Therefore, it is straightforward to see that $\mathcal{P}_1$ can also play a strategy $\lambda_1^{D_i} \in \Lambda_1^{PF}$ for a DAG D_i resulting of some merges on T. However, while this would be a valid strategy for $\mathcal{P}_1$, we have to prove that it is

still a winning one, for the same initial credit v_0 as λ_1^T. Precisely, we claim that $\forall\, i \geq 0$, we have that $\lambda_1^{D_i}$ is winning for v_0.

We show it by induction on D_i. The base case is trivial as $D_0 = T$: the strategy λ_1^T is winning for v_0 by definition. Our induction hypothesis is that our claim is valid for D_{i-1}, and we now prove it for D_i, by contradiction. Let $\varsigma_1, \varsigma_2 \in Q_{i-1}(j)$ be the merged nodes, for some level j of D_{i-1}. Suppose $\lambda_1^{D_i}$ is not winning for v_0. Thus there exists a finite path ζ of the pebble in D_i, which corresponds to a strategy $\lambda_2^{D_i} \in \Lambda_2^{PF}$ of $\mathcal{P}_2$, such that it achieves a negative value on at least one dimension m, $1 \leq m \leq k$. We have that $(v_0 + \mathsf{EL}(\zeta))\,(m) < 0$. We aim to find a similar path η in D_{i-1} such that $\mathsf{EL}(\eta) \leq \mathsf{EL}(\zeta)$, thus yielding contradiction, as it would witness that $\lambda_1^{D_{i-1}}$ is not winning for v_0.

We denote by ς_m the father of ς_2 in D_{i-1}. The only edge added by the merge operation is $(\varsigma_\mathsf{m}, \varsigma_1)$. Obviously, if ζ does not involve this edge, then we can take $\eta = \zeta$ and immediately obtain contradiction. Thus, we can decompose the witness path

$$\zeta = \alpha(1)\,\varsigma_\mathsf{m}\varsigma_1\,\beta(1)\,\circlearrowleft\,\alpha(2)\,\varsigma_\mathsf{m}\varsigma_1\,\beta(2)\,\circlearrowleft\,\ldots\,\circlearrowleft\,\alpha(q)\,\varsigma_\mathsf{m}\varsigma_1\,\xi,$$

for some $q \geq 1$ such that for all $1 \leq p \leq q$, we have that

- $\alpha(p), \beta(p), \xi \in (Q_i \cup \{\circlearrowleft\})^*$ are valid paths of the pebble in D_i (and D_{i-1});
- they do not involve edge $(\varsigma_\mathsf{m}, \varsigma_1)$, i.e., $\{\varsigma_\mathsf{m}\varsigma_1\} \not\subseteq \alpha(p), \beta(p), \xi$;
- $\beta(p) \cap \big(\mathsf{Anc}_{D_i}(\varsigma_\mathsf{m}) \setminus \mathsf{Anc}_{D_{i-1}}(\varsigma_1)\big) = \emptyset$, $\mathsf{Last}(\beta(p))$ is a leaf and it is the case that $\mathsf{oea}(\mathsf{Last}(\beta(p))) \in \mathsf{Anc}_{D_i}(\varsigma_\mathsf{m})$.

Intuitively, ζ is split into several parts in regard to q, the number of times it takes the added edge $(\varsigma_\mathsf{m}, \varsigma_1)$. Each time, this transition is preceded by some path α. It is then followed by some path β where all visited ancestors of ς_m were already ancestors of ς_1 in D_{i-1} (thus, β paths can be kept in η). Finally, after the q-th transition $\varsigma_\mathsf{m}\varsigma_1$ is taken, the path ζ ends with a finite sub-path ξ.

We define the witness path η in D_{i-1} as $\eta = \kappa(1)\beta(1)\,\circlearrowleft\,\kappa(2)\beta(2)\,\circlearrowleft\,\ldots\,\circlearrowleft\,\kappa(q)\xi$, with the following transformation of sub-paths $\alpha(p)\,\varsigma_\mathsf{m}\varsigma_1$:

- $\kappa(1) = r \rightsquigarrow_{D_{i-1}} \varsigma_1$,
- $\forall\, 2 \leq p \leq q, \kappa(p) = \mathsf{oea}(\mathsf{Last}(\beta(p-1))) \rightsquigarrow_{D_{i-1}} \varsigma_1$,

where $\rightsquigarrow_{D_{i-1}}$ denotes a valid path in D_{i-1}. Note that given preceding definitions, this indeed constitutes a valid path in D_{i-1}. We have to prove that $\mathsf{EL}(\eta) \leq \mathsf{EL}(\zeta)$. We have

$$\mathsf{EL}(\eta) = \sum_{1 \leq p \leq q} \mathsf{EL}(\kappa(p)) + \sum_{1 \leq p \leq q-1} \mathsf{EL}(\beta(p)) + \mathsf{EL}(\xi),$$

and

$$\mathsf{EL}(\zeta) = \sum_{1 \leq p \leq q} \mathsf{EL}(\alpha(p)\, \varsigma_{\mathsf{m}} \varsigma_1) + \sum_{1 \leq p \leq q-1} \mathsf{EL}(\beta(p)) + \mathsf{EL}(\xi).$$

Thus, it remains to show that

$$\sum_{1 \leq p \leq q} \mathsf{EL}(\kappa(p)) \leq \sum_{1 \leq p \leq q} \mathsf{EL}(\alpha(p)\, \varsigma_{\mathsf{m}} \varsigma_1).$$

In particular, we claim that for all $1 \leq p \leq q$, $\mathsf{EL}(\kappa(p)) \leq \mathsf{EL}(\alpha(p)\, \varsigma_{\mathsf{m}} \varsigma_1)$. Indeed, notice that $\kappa(p)$ and $\alpha(p)$ share their starting and ending nodes and that $\alpha(p)$ contains a finite number of pebble cycles. Let ϑ denote the common starting node of both $\kappa(p)$ and $\alpha(p)$. Applying Lemma 5.8 on $\alpha(p)$, we can eliminate cycles one at a time, without ever increasing the energy level, and obtain a path $\vartheta \rightsquigarrow_{D_i} \varsigma_{\mathsf{m}} \varsigma_1$ such that $\mathsf{EL}(\vartheta \rightsquigarrow_{D_i} \varsigma_{\mathsf{m}} \varsigma_1) \leq \mathsf{EL}(\alpha(p))$. Since $\varsigma_1 \simeq \varsigma_2$, we have by Lemma 5.7 that $\mathsf{EL}(\vartheta \rightsquigarrow_{D_i} \varsigma_{\mathsf{m}} \varsigma_1) = \mathsf{EL}(\vartheta \rightsquigarrow_{D_{i-1}} \varsigma_{\mathsf{m}} \varsigma_2) = \mathsf{EL}(\vartheta \rightsquigarrow_{D_{i-1}} \varsigma_1)$, implying the claim.

Consequently, we obtain $\mathsf{EL}(\eta) \leq \mathsf{EL}(\zeta)$, which witnesses that D_{i-1} was not winning. This contradicts our induction hypothesis and concludes our proof that for all $0 \leq i \leq n$, $\lambda_1^{D_i}$ is winning for v_0.

(ii) Second, the merge operation only prunes some parts of the tree T, without ever adding any new state, and added edges are on existing successive levels. Therefore, each D_i has noticeably the same depth l.

(iii) Third, the last DAG of the sequence, D_n, is such that for all level j, for all $\varsigma_1, \varsigma_2 \in Q_n(j)$, we have $(\varsigma_1 \neq \varsigma_2) \Rightarrow (\varsigma_1 \not\simeq \varsigma_2)$. Therefore the width of this DAG is bounded by the number of possible non-equivalent nodes. Recall that two nodes are equivalent if they have the same labels, i.e., they represent the same state of the game and are marked with exactly the same energy level vector. Since the maximal change in energy level on an edge is W, and the depth of the DAG is bounded by $l = 2^{(d-1)\cdot|S|} \cdot (W \cdot |S| + 1)^{c \cdot k^2}$ thanks to Lemma 5.4,

we have possible vectors in $\{-l \cdot W, -l \cdot W + 1, \ldots, l \cdot W - 1, l \cdot W\}^k$ for each state. Consequently, the width of D_n is bounded by

$$|S| \cdot (2 \cdot l \cdot W + 1)^k = |S| \cdot \left(2^{d \cdot |S|} \cdot (W \cdot |S| + 1)^{c \cdot k^2} \cdot W + 1\right)^k,$$

which is still single exponential. □

5.2 Single Exponential Lower Bound

In the next lemma, we show that the upper bound is tight in the sense that there exist families of games which require exponential memory (in the number of dimensions), even for the simpler case of multi energy objectives without parity and weights in $\{-1, 0, 1\}$ (Fig. 5.4). Note that for one-dimension energy parity, it was shown in [CD12] that exponential memory (in the encoding of weights) may be necessary (cf. Sect. 4.3.3).

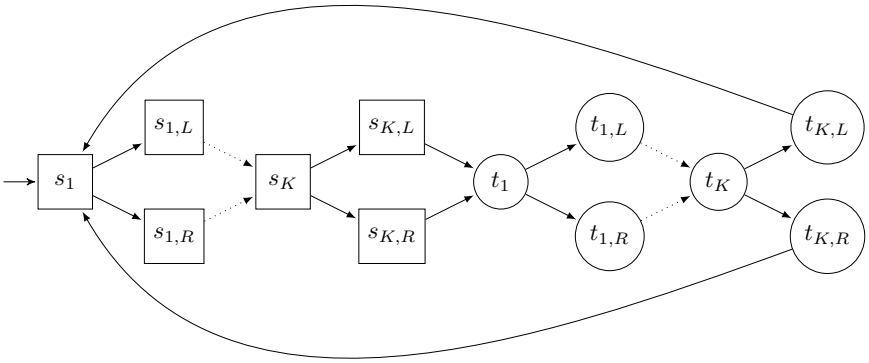


Figure 5.4: Family of games requiring exponential memory.

Lemma 5.9. *There exists a family of multi energy games* $(G(K))_{K \geq 1} = (S_1, S_2, E, k = 2 \cdot K, w : E \to \{-1, 0, 1\}^k)$ *such that for any initial credit,* $\mathcal{P}_1$ *needs exponential memory to win.*

The idea is the following: in the example of Fig. 5.4, if $\mathcal{P}_1$ does not remember the exact choices of $\mathcal{P}_2$ (which requires an exponential size Moore machine),

there will exist some sequence of choices of $\mathcal{P}_2$ such that $\mathcal{P}_1$ cannot counteract a decrease in energy. Thus, by playing this sequence long enough, $\mathcal{P}_2$ can force $\mathcal{P}_1$ to lose, whatever his initial credit is.

Proof. We define a family of games $(G(K))_{K\geq 1}$ which is an assembly of $k = 2 \cdot K$ gadgets, the first K belonging to $\mathcal{P}_2$, and the remaining K belonging to $\mathcal{P}_1$ (Fig. 5.4). Precisely, we have $|S_1| = |S_2| = 3 \cdot K$, $|S| = |E| = 6 \cdot K = 3 \cdot k$ (linear in k), $k = 2 \cdot K$, and w defined as:

$$\begin{aligned}
&\forall\, 1 \leq i \leq K, w((\circ, s_i)) = w((\circ, t_i)) = (0, \dots, 0),\\
&\qquad w((s_i, s_{i,L})) = -w((s_i, s_{i,R})) = w((t_i, t_{i,L})) = -w((t_i, t_{i,R})),\\
&\qquad \forall\, 1 \leq j \leq k,\ w((s_i, s_{i,L}))(j) = \begin{cases} 1 \text{ if } j = 2 \cdot i - 1 \\ -1 \text{ if } j = 2 \cdot i \\ 0 \text{ otherwise} \end{cases},
\end{aligned}$$

where $\circ$ denotes any valid predecessor state.

There exists a winning strategy λ_1^{exp} for $\mathcal{P}_1$, for initial credit $v_0^{exp} = (1, \dots, 1)$. Indeed, for any strategy of $\mathcal{P}_2$, for any state t_i belonging to $\mathcal{P}_1$, it suffices to play the *opposite* choice as $\mathcal{P}_2$ made on its last visit of s_i to maintain at all times an energy vector which is positive on all dimensions. This strategy thus requires to remember the last choice of $\mathcal{P}_2$ in all gadgets, which means $\mathcal{P}_1$ needs K bits to encode these decisions. Thus, this winning strategy is described by a Moore machine containing $2^K = 2^{\frac{k}{2}}$ states, which is exponential in the number of dimensions k.

We claim that, for any initial credit v_0, there exists no winning strategy λ_1 that can be described with less than 2^K states and prove it by contradiction. Suppose $\mathcal{P}_1$ plays according to such a strategy λ_1. Then there exists some $1 \leq x \leq K$ such that $\lambda_1(s_1 \dots s_x s_{x,L} \dots t_x) = \lambda_1(s_1 \dots s_x s_{x,D} \dots t_x)$, i.e., $\mathcal{P}_1$ chooses the same action in t_x against both choices of the adversary. Suppose that $\mathcal{P}_1$ chooses to play $t_{x,L}$ in both cases, that is $\lambda_1(s_1 \dots s_x s_{x,L} \dots t_x) = \lambda_1(s_1 \dots s_x s_{x,D} \dots t_x) = t_{x,L}$. By playing $s_{x,L}$, $\mathcal{P}_2$ can force a decrease of the energy vector by 2 on dimension $2 \cdot x$ every visit in gadget x. Similarly, if the strategy of $\mathcal{P}_1$ is to play $t_{x,R}$, $\mathcal{P}_2$ wins by choosing to play $s_{x,R}$ as dimension $2 \cdot x - 1$ decreases by 2 every visit. Therefore, whatever the finite initial vector

of $\mathcal{P}_1$, $\mathcal{P}_2$ can enforce a negative dimension by playing long enough. This contradicts the fact that λ_1 is winning and concludes our proof that exponential memory is necessary for this simple family of games $(G(K))_{K\geq 1}$. □

5.3 Wrap-up

We summarize our results on memory bounds in Theorem 5.10.

Theorem 5.10 (Optimal memory bounds). *The following assertions hold: (1) In multi energy parity games, if there exists a winning strategy, then there exists a finite-memory winning strategy. (2) In multi energy parity and multi mean-payoff games, if there exists a finite-memory winning strategy, then there exists a winning strategy with at most exponential memory. (3) There exists a family of multi energy games (without parity) with weights in* $\{-1, 0, 1\}$ *where all winning strategies require at least exponential memory.*

Proof. Thanks to Lemma 4.10, we have equivalence between finite-memory winning for multi energy and multi mean-payoff games. The rest follows from straigthforward application of Lemma 5.1, Lemma 5.5, Lemma 5.6, and finally Lemma 5.9. □

Remark 5.11. We do not distinguish infimum and supremum variants of the mean-payoff objective in this context as they coincide for finite-memory strategies, as discussed in Sect. 4.3. ◁

CHAPTER 6

Symbolic Synthesis Algorithm

Algorithm ◇ Correctness and Completeness ◇ Applicability

We present a *symbolic* algorithm (in the sense of [DR10], i.e., using a compact antichain representation of sets by their minimal elements) to compute a finite-memory winning strategy, if one exists, for multi energy parity games.

Our algorithm is parameterized by the range of energy levels to consider during its execution. So, we can use it in an *incremental approach*: first, we search for finite-memory winning strategies within a small range, and increment the range only when necessary. We also establish a bound on the maximal range to consider which ensures completeness of the incremental approach.

In the worst case the algorithm requires exponential time. Since exponential size memory is required (and the decision problem is coNP-complete), the worst case exponential bound can be considered as *optimal*. Moreover, as our algorithm is symbolic and incremental, in most relevant problems in practice, it is expected to be efficient. We mention an implementation by Bohy et al. [BBFR13].

Our algorithm was developed and presented in joint work with Chatterjee and Raskin [CRR12a, CRR12b, CRR13].

6.1 Algorithm

We present a *symbolic*, *incremental* and *optimal* algorithm to synthesize a finite-memory winning strategy in a MEG.[1] This algorithm outputs a (set of) winning initial credit(s) and a derived finite-memory winning strategy (if one exists) which is exponential in the worst-case. Its running time is at most exponential. So our symbolic algorithm can be considered (worst-case) optimal in the light of the results of Chap. 5.

This algorithm computes the greatest fixed point of a monotone operator that defines the sets of winning initial (vectors of) credits for each state of the game. As those sets are upward-closed, they are symbolically represented by their minimal elements. To ensure convergence, the algorithm considers only credits that are below some *threshold*, noted $\mathbb{C}$. This is without giving up completeness because, as we show in the following, for a game $G = (S_1, S_2, E, k, w)$, it is sufficient to take the value $2 \cdot l \cdot W$ for $\mathbb{C}$, where l is the bound on the depth of epSCTs obtained in Lemma 5.4 and W is the largest absolute value of weights used in the game. We also show how to extract a deterministic output Moore machine representing a corresponding winning strategy from this set of minimal winning initial credits and how to obtain an *incremental* algorithm by increasing values for the threshold $\mathbb{C}$ starting from small values.

A controllable predecessor operator. Let $G = (S_1, S_2, E, k, w)$ be a MEG, $\mathbb{C} \in \mathbb{N}$ be a constant, and $U(\mathbb{C})$ be the set $(S_1 \cup S_2) \times \{0, 1, \ldots, \mathbb{C}\}^k$. Let $\mathcal{U}(\mathbb{C}) = 2^{U(\mathbb{C})}$, i.e., the powerset of $U(\mathbb{C})$, and the operator $\mathsf{Cpre}_{\mathbb{C}} \colon \mathcal{U}(\mathbb{C}) \to \mathcal{U}(\mathbb{C})$ be defined as follows:

$$\mathcal{E}(V) = \{(s_1, e_1) \in U(\mathbb{C}) \mid s_1 \in S_1 \wedge \exists\, (s_1, s) \in E, \exists\, (s, e_2) \in V : e_2 \leq e_1 + w(s_1, s)\},$$

$$\mathcal{A}(V) = \{(s_2, e_2) \in U(\mathbb{C}) \mid s_2 \in S_2 \wedge \forall\, (s_2, s) \in E, \exists\, (s, e_1) \in V : e_1 \leq e_2 + w(s_2, s)\},$$

$$\mathsf{Cpre}_{\mathbb{C}}(V) = \mathcal{E}(V) \cup \mathcal{A}(V). \tag{6.1}$$

[1] Note that the symbolic algorithm can be applied to MEPGs and MMPPGs after removal of the parity condition by applying the construction of Lemma 5.5.

Intuitively, $\mathsf{Cpre}_{\mathbb{C}}(V)$ returns the set of energy levels from which $\mathcal{P}_1$ can force an energy level in V in one step.

Example 6.1. We illustrate the execution of operator $\mathsf{Cpre}_{\mathbb{C}}$ on a two-dimension game in Fig. 6.1. We consider the aggregation of winning credits on state s (sets in red), assuming the sets of winning credits for successor states s' (sets in blue) and s'' (sets in orange) are already known. It is assumed that edges from s have weight zero for the sake of readability.

If s belongs to $\mathcal{P}_1$, then he can choose the most profitable situation for him. Hence, we take the *union* of the sets of winning credits for s' and s''. If s belongs to $\mathcal{P}_2$, then $\mathcal{P}_1$ must be able to win from both state s' and state s''. Indeed, he cannot predict which one will be chosen by $\mathcal{P}_2$. Therefore, we take the *intersection* of the sets. Observe that given a state, the corresponding set of winning credits is a union of upper closed sets. This union can be represented efficiently by the minimal elements of these upper closed sets, as discussed in the following.

The green lines represent the upper bound $\mathbb{C}$ that encloses the space of considered winning credits. This bound guarantees convergence of the fixed point computation that is the core of our algorithm. ◁

The operator $\mathsf{Cpre}_{\mathbb{C}}$ is $\subseteq$-monotone over the complete lattice $\mathcal{U}(\mathbb{C})$, and so there exists a *greatest fixed point* for $\mathsf{Cpre}_{\mathbb{C}}$ in the lattice $\mathcal{U}(\mathbb{C})$, denoted by $\mathsf{Cpre}^*_{\mathbb{C}}$. As usual, the greatest fixed point of the operator $\mathsf{Cpre}_{\mathbb{C}}$ can be computed by successive approximations as the last element of the following finite $\subseteq$-descending chain. We define the algorithm CpreFP that computes this greatest fixed point:

$$U_0 = U(\mathbb{C}),\ U_1 = \mathsf{Cpre}_{\mathbb{C}}(U_0),\ \ldots,\ U_n = \mathsf{Cpre}_{\mathbb{C}}(U_{n-1}) = U_{n-1}. \tag{6.2}$$

The set U_i contains all the energy levels that are sufficient to maintain the energy positive in all dimensions for i steps. Note that the length of this chain can be bounded by $|U(\mathbb{C})|$ and the time needed to compute each element of the chain can be bounded by a polynomial in $|U(\mathbb{C})|$. As a consequence, we obtain the following lemma.

Lemma 6.2. *Let $G = (S_1, S_2, E, k, w)$ be a multi energy game and $\mathbb{C} \in \mathbb{N}$ be a constant. Then $\mathsf{Cpre}^*_{\mathbb{C}}$ can be computed in time bounded by a polynomial in $|U(\mathbb{C})|$, i.e., an exponential in the size of G.*

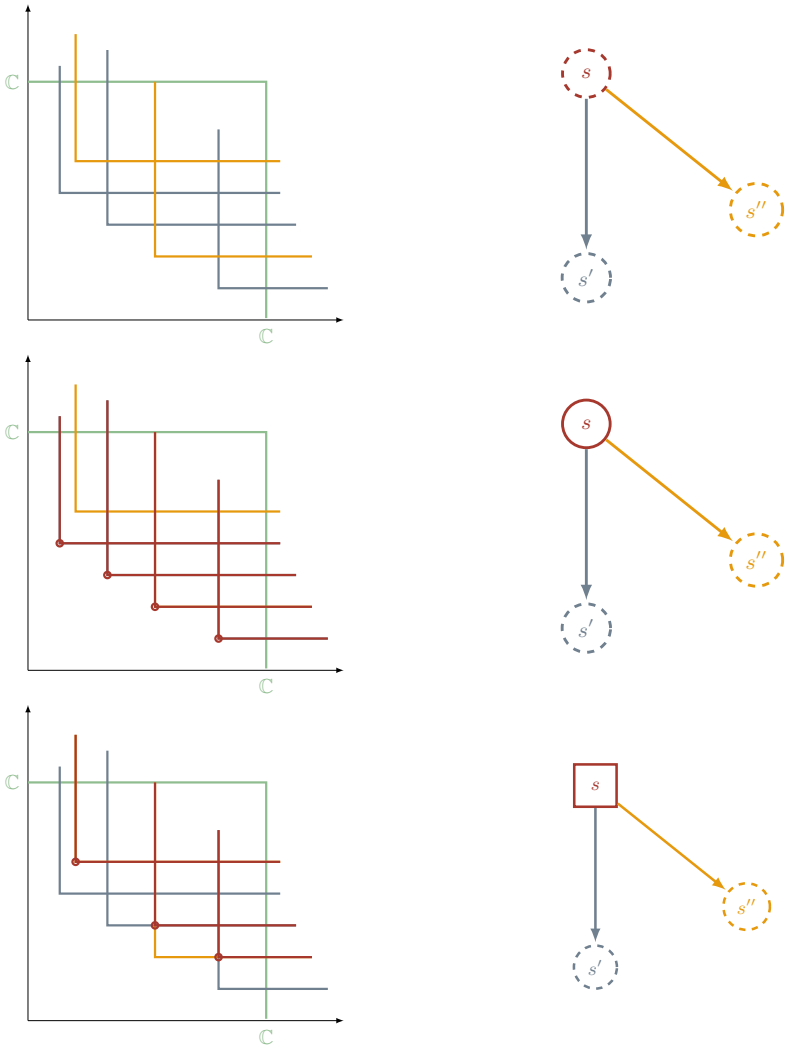


Figure 6.1: Aggregation of winning credits by application of operator $\mathsf{Cpre}_{\mathbb{C}}$, depending on the state possessor.

Symbolic representation. To define a symbolic representation of the sets manipulated by the $\mathsf{Cpre}_{\mathbb{C}}$ operator, we exploit the following partial order: let $(s, e), (s', e') \in U(\mathbb{C})$, we define

$$(s, e) \preceq (s', e') \text{ iff } s = s' \text{ and } e \leq e'.$$

A set $V \in \mathcal{U}(\mathbb{C})$ is *closed* if for all $(s,e),(s',e') \in U(\mathbb{C})$, if $(s,e) \in V$ and $(s,e) \preceq (s',e')$, then $(s',e') \in V$. By definition of $\mathsf{Cpre}_{\mathbb{C}}$, we get the following property.

Lemma 6.3. *All sets U_i in eq. (6.2) are closed for $\preceq$.*

Therefore, all sets U_i in the descending chain of eq. (6.2) can be symbolically represented by their minimal elements $\mathsf{Min}_{\preceq}(U_i)$ which is an antichain of elements for $\preceq$. Even if the largest antichain can be exponential in G, this representation is, in practice, often much more efficient, even for small values of the parameters. For example, with $\mathbb{C} = 4$ and $k = 4$, we have that the cardinality of a set can be as large as $|U_i| \leq 625$ whereas the size of the largest antichain is bounded by $|\mathsf{Min}_{\preceq}(U_i)| \leq 35$. Antichains have proved to be very efficient data structures for many applications: see for example [DDHR06, ACH+10, DR10], or Ducobu's [Duc13] and Maquet's [Maq11] theses. Therefore, our algorithm is expected to have good performance in practice.

6.2 Correctness and Completeness

The following two lemmas relate the greatest fixed point $\mathsf{Cpre}^*_{\mathbb{C}}$ and the existence of winning strategies for $\mathcal{P}_1$ in a multi energy game G. We start with the correctness of the symbolic algorithm.

Lemma 6.4 (Correctness). *Let $G = (S_1, S_2, E, k, w)$ be a multi energy game, $s_{\mathsf{init}} \in S$ an initial state, and $\mathbb{C} \in \mathbb{N}$ a constant. If there exists $(c_1, \ldots, c_k) \in \mathbb{N}^k$ such that $(s_{\mathsf{init}}, (c_1, \ldots, c_k)) \in \mathsf{Cpre}^*_{\mathbb{C}}$, then $\mathcal{P}_1$ has a winning strategy in G for initial credit $(c_1, \ldots, c_k)$ and the memory of $\mathcal{P}_1$ can be bounded by $|\mathsf{Min}_{\preceq}(\mathsf{Cpre}^*_{\mathbb{C}})|$ (the size of the antichain of minimal elements in the fixed point).*

Given the set of winning initial credits output by CpreFP, it is straightforward to derive a corresponding winning strategy of at most exponential size. Indeed, for winning initial credit $\overline{c} \in \mathbb{N}^k$, we build a deterministic output Moore machine which (i) states are the minimal elements of the fixed point (antichain at most exponential in G), (ii) initial state is any element (t,u) among them such that $t = s_{\mathsf{init}}$ and $u \leq \overline{c}$, (iii) update function maintains an accurate energy level in the memory, and (iv) next-action function prescribes an action that ensures remaining in the fixed point.

Proof. We denote by $\overline{c}$ the k-dimension credit vector $(c_1, \ldots, c_k)$. W.l.o.g. we assume that states of G alternate between positions of $\mathcal{P}_1$ and positions of $\mathcal{P}_2$ (otherwise, we split needed edges by introducing dummy states). From $\mathsf{Cpre}^*_{\mathbb{C}}$, we construct a deterministic output Moore machine $\mathcal{M} = (\mathsf{Mem}, \mathsf{m}_0, \alpha_{\mathsf{u}}, \alpha_{\mathsf{n}})$ which respects the following definitions:[2]

- $\mathsf{Mem} = \mathsf{Min}_{\preceq}\{(t,u) \in S_1 \times \{0 \ldots \mathbb{C}\}^k \mid (t,u) \in (\mathsf{Cpre}^*_{\mathbb{C}})\}$. The set of states of the machine is the antichain of $\preceq$-minimal elements that belong to $\mathcal{P}_1$ in the fixed point. Note that the length of this antichain is bounded by an exponential in the size of the game.
- m_0 is any element (t,u) in Mem such that $t = s_{\mathsf{init}}$ and $u \leq \overline{c}$. Note that such an element is guaranteed to exist as $(s_{\mathsf{init}}, \overline{c}) \in \mathsf{Cpre}^*_{\mathbb{C}}$.
- For all $(t,u) \in \mathsf{Mem}$, we define $\alpha_{\mathsf{n}}((t,u))$ by choosing any element $(t,t') \in E$ such that there exists $(t',u') \in \mathsf{Cpre}^*_{\mathbb{C}}$ with $u' = u + w(t,t')$. Such an element is guaranteed to exist by definition of $\mathsf{Cpre}_{\mathbb{C}}$ and the fact that $(t,u) \in \mathsf{Cpre}^*_{\mathbb{C}}$.
- $\alpha_{\mathsf{u}} \colon \mathsf{Mem} \times ((S_2 \times S) \cap E) \to \mathsf{Mem}$ is any partial function that respects the following constraint: if $\alpha_{\mathsf{n}}((t,u)) = (t,t')$ then $\alpha_{\mathsf{u}}((t,u),(t',t''))$ is defined for any $(t',t'') \in E$ and can be chosen to be equal to any (t'',u'') such that $u'' \leq u + w(t,t') + w(t',t'')$, and such an u'' is guaranteed to exist by definition of $\mathsf{Cpre}_{\mathbb{C}}$ and because $\mathsf{Cpre}^*_{\mathbb{C}}$ is a fixed point.

Now, let us prove that for any initial prefix $s_0 s_1 \ldots s_{2n}$ of even length in G, which is compatible with $\mathcal{M}$, we have that $\overline{c} + \mathsf{EL}(s_0 s_1 \ldots s_{2n-1}) \geq 0$ and that $\overline{c} + \mathsf{EL}(s_0 s_1 \ldots s_{2n}) \geq 0$. To establish this property, we first prove the following property by induction on n: $\overline{c} + \mathsf{EL}(s_0 s_1 \ldots s_{2n}) \geq u$ where u is the energy level of the label of the state reached after reading the prefix $s_0 s_1 \ldots s_{2n}$ with the Moore machine $\mathcal{M}$. Base case $n = 0$ is trivial. Induction: assume that the property is true for $n-1$, and let us establish it for n. By induction hypothesis, we have that $\overline{c} + \mathsf{EL}(s_0 s_1 \ldots s_{2(n-1)}) \geq u$ where u is the energy level of the label of state m that is reached after reading $s_0 s_1 \ldots s_{2(n-1)}$ with the Moore machine. Now, assume that $\alpha_{\mathsf{n}}(\mathsf{m}) = (t,t')$. So, $s_{2(n-1)} = t$ and the choice of $\mathcal{P}_1$ is to play (t,t'). So, $s_{2(n-1)+1} = t'$. Now for all possible choices (t',t'') of $\mathcal{P}_2$, we know by definition of $\mathcal{M}$ that the energy level u'' that labels the state $\alpha_{\mathsf{u}}(\mathsf{m},(t',t''))$

[2] We slightly differ from the usual notation to ease up the definition.

is $u'' \leq u + w(t, t') + w(t', t'')$, which establishes our property. Therefore, the strategy of $\mathcal{P}_1$ based on $\mathcal{M}$ is such that the energy always stays positive for initial credit $\bar{c}$, which concludes the proof. □

Completeness of the symbolic algorithm is guaranteed when a sufficiently large threshold $\mathbb{C}$ is used as established in the following lemma.

Lemma 6.5 (Completeness). *Let $G = (S_1, S_2, E, k, w)$ be a multi energy game in which all absolute values of weights are bounded by W. If $\mathcal{P}_1$ has a winning strategy in G from initial state $s_{\mathsf{init}} \in S$ and $T = (Q, R)$ is a self-covering tree for G of depth l, then $(s_{\mathsf{init}}, (\mathbb{C}, \ldots, \mathbb{C})) \in \mathsf{Cpre}^*_{\mathbb{C}}$ for $\mathbb{C} = 2 \cdot l \cdot W$.*

Remark 6.6. This algorithm is complete in the sense that if a winning strategy exists for $\mathcal{P}_1$, it outputs at least a winning initial credit (and the derived strategy) for $\mathbb{C} = 2 \cdot l \cdot W$. However, this is different from the *fixed initial credit problem*, which consists in deciding if a particular given credit vector is winning and is known to be EXPSPACE-hard by equivalence with deciding the existence of an infinite run in a Petri net given an initial marking [BJK10, FJLS11]. In general, there may exist winning credits that are incomparable to those captured by algorithm CpreFP.

More precisely, given a constant $\mathbb{C} \in \mathbb{N}$, the algorithm fully captures all the winning initial credits smaller than $(\mathbb{C}, \ldots, \mathbb{C})$. Indeed, the fixed point computation considers the whole range of initial credits up to the given constant exhaustively, and only removes credits if they do not suffice to win. By Lemma 6.5, it is moreover guaranteed that if an arbitrary winning initial credit exists, then there exists one in the range defined by the constant $\mathbb{C} = 2 \cdot l \cdot W$.

Nevertheless, since our algorithm works in exponential time while the problem of finding *all* the winning initial credits is EXPSPACE-hard, there may be some incomparable credits outside that range that are not captured by the algorithm (comparable credits are captured since we work with upper closed sets). Indeed, if our algorithm was able to compute exhaustively all winning credits in exponential time, this would induce that EXPTIME is equal to EXPSPACE. Notice that defining a class of games for which the algorithm CpreFP proves to be incomplete (in the sense that incomparable winning credits exist outside the region captured by constant $\mathbb{C} = 2 \cdot l \cdot W$) is an interesting open problem.

Besides, computation of the Pareto frontier (i.e., complete representation of all incomparable winning credits) was addressed by Abdulla et al. [AMSS13], partly based on our results. They show that this frontier is decidable, but complexity is left open. No lower bound is known other than the EXPSPACE-hardness already valid for the simpler fixed initial credit problem. ◁

Proof. To establish this property, we first prove that from the set of labels of T, we can construct a set f which is increasing for the operator $\mathsf{Cpre}_{\mathbb{C}}$, i.e., $\mathsf{Cpre}_{\mathbb{C}}(f) \supseteq f$, and such that $(s_{\mathsf{init}}, (\mathbb{C}, \ldots, \mathbb{C})) \in f$. We define f from $T = (Q, R)$ as follows. Let $C \in \mathbb{N}$ be the smallest non-negative integer such that for all $q \in Q$, with $\Theta(q) = (t, u)$, for all dimensions i, $1 \leq i \leq k$, we have that $u(i) + C \geq 0$. Integer C is bounded from above by $l \cdot W$ because on every path from the root to a leaf in T, every dimension is at most decreased l times by an amount bounded by W, and at the root all the dimensions are equal to 0. For any $q \in Q$, we denote by $\Theta(q) + C$ the label of q where the energy level has been increased by C in all the dimensions, i.e., if $\Theta(q) = (t, u)$ then $\Theta(q) + C = (t, u + (C, \ldots, C))$. Note that for all nodes in Q, the label is at most $l \cdot W$ and thus the shifted label remains under $\mathbb{C} = 2 \cdot l \cdot W$. Now, we define the set f as follows:

$$f = \{(t, u) \in U(\mathbb{C}) \mid \exists\, q \in Q,\ \Theta(q) + C \preceq (t, u)\}.$$

So, f is defined as the $\preceq$-closure of the set of labels in T shifted by C in all the dimensions.

First, note that $(s_{\mathsf{init}}, (\mathbb{C}, \ldots, \mathbb{C})) \in f$ because the label of the root in T is $(s_{\mathsf{init}}, (0, \ldots, 0))$. Second, let us show that $\mathsf{Cpre}_{\mathbb{C}}(f) \supseteq f$. Take any $(t, u) \in f$ and let us show that $(t, u) \in \mathsf{Cpre}_{\mathbb{C}}(f)$. We decompose the proof in two cases.

Case (A) $t \in S_1$. By definition of f, there exists $q \in Q$ such that $\Theta(q) + C \preceq (t, u)$. W.l.o.g. we can assume that q is not a leaf as otherwise there exists an ancestor q' of q such that $\Theta(q') \preceq \Theta(q)$ (recall the set is described by its minimal elements). By definition of T, there exists $(t, t') \in E$ and $q' \in Q$ such that $(q, q') \in R$ and $\Theta(q') = \Theta(q) + w(t, t')$. Let $(t', v) = \Theta(q') + C$. By definition of f, we have $(t', v) \in f$. By eq. (6.1), it follows that $(t, u) \in \mathsf{Cpre}_{\mathbb{C}}(f)$.

Case (B) $t \in S_2$. By definition of f, there exists $q \in Q$ such that $\Theta(q) + C \preceq (t, u)$. Again, w.l.o.g. we can assume that q is not a leaf as otherwise there exists an ancestor q' of q such that $\Theta(q') \preceq \Theta(q)$. By definition of T, for all

$(t, t') \in E$, there is $q' \in Q$ such that $(q, q') \in R$ and $\Theta(q') = \Theta(q) + w(t, t')$. Let $(t', v) = \Theta(q') + C$. By definition of f, we have $(t', v) \in f$. By eq. (6.1), it follows that $(t, u) \in \mathsf{Cpre}_{\mathbb{C}}(f)$.

Now, let us show that $f \subseteq \mathsf{Cpre}^*_{\mathbb{C}}$. This is a direct consequence of the monotonicity of $\mathsf{Cpre}_{\mathbb{C}}$: it is well known that for any monotone function on a complete lattice, its greatest fixed point is equal to the least upper bound of all post-fixed points (points e such that $e \subseteq \mathsf{Cpre}_{\mathbb{C}}(e)$), i.e., $\mathsf{Cpre}^*_{\mathbb{C}} = \bigcup\{e \mid e \subseteq \mathsf{Cpre}_{\mathbb{C}}(e)\} \supseteq f$. As $(s_{\mathsf{init}}, (\mathbb{C}, \dots, \mathbb{C})) \in f$, that concludes the proof. □

Remark 6.7. The exponential bound on memory, obtained in Lemma 5.6, can also be derived from the Moore machine construction of Lemma 6.4 as this method is complete according to Lemma 6.5. Still, the DAG construction of Lemma 5.6 is interesting in its own right, and introduces the concept of node merging, which is a cornerstone of the symbolic algorithm correctness, while transparent in its use. ◁

6.3 Applicability

Incrementality. While the threshold $2 \cdot l \cdot W$ is sufficient, it may be the case that $\mathcal{P}_1$ can win the game even if its energy level is bounded above by some smaller value. So, in practice, we can use Lemma 6.4, to justify an incremental algorithm that first starts with small values for the parameter $\mathbb{C}$ and stops as soon as a winning strategy is found or when the value of $\mathbb{C}$ reaches the threshold $2 \cdot l \cdot W$ and no winning strategy has been found.

Application of the symbolic algorithm to MEPGs and MMPGs. Using the reduction of Lemma 5.5 that allows us to remove the parity condition, and the equivalence between multi energy games and multi mean-payoff games for finite-memory strategies (given by Lemma 4.10), along with Lemma 6.2 (complexity), Lemma 6.4 (correctness) and Lemma 6.5 (completeness), we obtain the following result.

Theorem 6.8 (Symbolic and incremental synthesis algorithm). *Let G_p be a multi energy (resp. multi mean-payoff) parity game and s_{init} an initial state. Algorithm* CPREFP *is a symbolic and incremental algorithm that synthesizes a winning strategy in G_p of at most exponential size memory, if a winning*

(resp. finite-memory winning) strategy exists. In the worst-case, the algorithm CpreFP *takes exponential time.*

Proof. The correctness and completeness for algorithm CpreFP on multi energy games are respectively given by Lemma 6.4 and Lemma 6.5. Extension to mean-payoff games (under finite memory) is given by Lemma 4.10, whereas the parity condition can be encoded as energy thanks to Lemma 5.5. Exponential worst-case complexity of the algorithm CpreFP is induced by Lemma 6.2. □

Integration in synthesis tools. Following our related publications [CRR12a, CRR12b, CRR13], our results on strategy synthesis have been used in the synthesis tool Acacia+. This tool originally handled the synthesis of controllers for specifications expressed in LTL (Linear Temporal Logic, a classical formalism for formal specifications [Pnu77]) using antichain-based algorithms [BBF+12] and has recently been extended to the synthesis from LTL specifications with mean-payoff objectives [BBFR13]. The addition of multi mean-payoff objectives to LTL specifications provides a convenient way to enforce that synthesized controllers also satisfy some reasonable behavior from a quantitative standpoint, such as minimizing the number of unsollicited grants in a client-server architecture with prioritized clients. Numerous practical applications may benefit from this multi-dimension framework.

The authors present an approach in which the corresponding synthesis problem ultimately reduces to strategy synthesis on a multi energy game [BBFR13, Theorem 26]. Their implementation uses fixed point computations similar to eq. (6.2) and has proved efficient (considering the complexity of the problem) in practice. It uses antichains to provide a compact representation of upper-closed sets and implements the incremental approach proposed before (regarding the constant $\mathbb{C}$). In practical benchmarks, winning strategies can generally be found for rather small values of $\mathbb{C}$. Hence, the incremental approach overcomes the need to compute up to the exponential theoretical bound $\mathbb{C} = 2 \cdot l \cdot W$ in many cases. Sample benchmarks and experiments can be found in [BBFR13], and the tool can be used online.[3]

[3] http://lit2.ulb.ac.be/acaciaplus/

CHAPTER 7

Trading Finite Memory for Randomness

Introduction ◇ Energy Games ◇ Multi Mean-Payoff (Parity) Games ◇ Single Mean-Payoff Parity Games ◇ Wrap-up

We consider several classes of multi-dimension quantitative games and study when pure finite-memory strategies of $\mathcal{P}_1$ can be traded for conceptually much simpler *randomized* memoryless strategies.

We show that for energy objectives, randomization is not helpful even with only one player, as energy objectives are similar in spirit with safety objectives. It is no more useful in two-player multi mean-payoff games.

However, randomized memoryless strategies suffice for one-player games with multi mean-payoff parity objectives. For the important special case of mean-payoff parity objectives (conjunction of a single mean-payoff objective and a parity objective), we show that in two-player games, finite-memory strategies can effectively be replaced by randomized memoryless strategies.

These results stem from collaboration with Chatterjee and Raskin [CRR12a, CRR12b, CRR13].

7.1 Introduction

We answer the fundamental question regarding the trade-off of memory for randomness in strategies: we study on which kind of games $\mathcal{P}_1$ can replace a pure finite-memory winning strategy by an equally powerful, yet conceptually simpler, randomized memoryless one and discuss how memory is encoded into probability distributions. We consider both *expected value semantics* and *almost-sure semantics*.

Note that we do not consider wider strategy classes (e.g., randomized finite-memory), nor do we allow randomization for $\mathcal{P}_2$ (which on most cases is dispensable anyway). Indeed, we aim at a *better understanding of the underlying mechanics of memory and randomization*, in order to provide alternative strategy representations of practical use; not exploration of more complex games with wider strategy classes (Lemma 7.12 shows a glimpse of it).

	Multi energy and energy parity	Multi MP (parity)	MP parity
one-player	×	√	√
two-player	×	×	√

Table 7.1: When pure finite memory for $\mathcal{P}_1$ can be traded for randomized memorylessness.

We present an overview of our results in Tab. 7.1 and summarize them in Theorem 7.11. Note that we do not consider the opposite implication, i.e., does there always exist a way of encoding a randomized memoryless strategy into an equivalent finite-memory one. In general, this is not the case even for classes of games where we can trade memory for randomness, and it can easily be witnessed on the one-player multi mean-payoff game depicted in Fig. 7.1. Consider the *expectation semantics*. Expectation $(1, 1)$ is achievable with a simple uniform distribution while it is not achievable with a pure, arbitrary high memory strategy (even infinite).

We break down these results into three sections: energy games, multi mean-payoff (parity) games, and single mean-payoff parity games. We start by considering energy games.

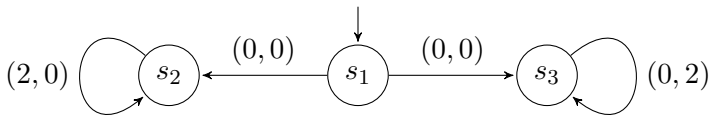


Figure 7.1: Randomization can replace memory, but not the opposite.

Remark 7.1. Since we consider finite-memory strategies, induced Markov chains are finite and the usual argument holds to obtain that infimum and supremum variants of the mean-payoff function coincide for worst-case and expected value problems. In the following, we state our results for the infimum definition. ◁

7.2 Energy Games

Randomization is not helpful for energy objectives, even in one-player games. The proof argument is obtained from the intuition that energy objectives are similar in spirit to safety objectives. This was already illustrated in Ex. 2.28.

Lemma 7.2. *Randomization is not helpful for almost-sure winning in one-player and two-player energy, multi energy, energy parity and multi energy parity games: if there exists a finite-memory randomized winning strategy, then there exists a pure winning strategy with the same memory requirements.*

Proof. Let G be a game fitted with an energy objective. Consider an almost-sure winning strategy λ_1. If there exists a single path π consistent with λ_1 that violates the energy objective, then there exists a finite prefix witness ρ sufficient to violate the energy objective. Moreover, as the finite prefix has positive probability (otherwise the play is not consistent), and the strategy λ_1 is almost-sure winning, it follows that no such path exists. In other words, λ_1 is a sure winning strategy. Since randomization does not help for sure semantics, it follows that randomization is not helpful for one-player and two-player energy, multi energy, energy parity and multi energy parity games. □

7.3 Multi Mean-Payoff (Parity) Games

Randomized memoryless strategies can replace pure finite-memory ones in the one-player multi mean-payoff parity case, but not in the two-player one, even without parity. We first note a useful link between satisfaction and expectation semantics for the mean-payoff objective.

Lemma 7.3. *Let $G = (S_1, S_2, E, k, w)$ be a game with initial state $s_{\mathsf{init}} \in S$ and mean-payoff objective $\phi = \mathsf{MeanInf}_G(\mu)$ for some threshold vector $\mu \in \mathbb{Q}^k$. Let $\lambda_1 \in \Lambda_1^F$ be a strategy of $\mathcal{P}_1$. If λ_1 is almost-sure winning for ϕ, then λ_1 also guarantees expectation at least μ for the mean-payoff function $\underline{\mathsf{MP}}$. The opposite does not hold.*

Proof. We first discuss the claimed implication. Suppose almost-sure winning is verified. For all strategy $\lambda_2 \in \Lambda_2$ of $\mathcal{P}_2$, the set of consistent plays of mean-payoff greater than or equal to μ has measure one, while the one of value strictly less than μ has measure zero, by definition. Therefore, the expected mean-payoff $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1,\lambda_2]}(\underline{\mathsf{MP}})$ is at least μ.

To show that the opposite does not hold, consider the simple one-player game depicted in Fig. 7.1. Let λ_1 be a simple coin flipping on s_1, i.e., $\lambda_1(s_1)(s_2) = 1/2$, $\lambda_1(s_1)(s_3) = 1/2$, $\lambda_1(s_2)(s_2) = 1$ and $\lambda_1(s_3)(s_3) = 1$. The expectation of this strategy is $v = (1, 1)$. Nevertheless, the probability of achieving mean-payoff at least v is $0 < 1$, which shows that the strategy is not almost-surely winning for objective $\mathsf{MeanInf}_G(v)$. □

The fundamental difference between energy and mean-payoff is that energy requires a property to be satisfied *at all times* (in that sense, it is similar to safety), while mean-payoff is a *limit* property. As a consequence, what matters here is the long-run frequencies of weights, not their order of appearance, as opposed to the energy case.

Lemma 7.4. *Pure finite-memory winning strategies can be traded for equally powerful randomized memoryless ones for one-player multi mean-payoff parity games, for both satisfaction and expectation semantics. For two-player games, randomized memoryless strategies are not as powerful, even limited to expectation semantics, no parity condition, and only two dimensions.*

For the one-player case, we extract the frequencies of visit for edges of the graph from the regular outcome that arises from the finite-memory strategy of $\mathcal{P}_1$. We build a randomized strategy with probability distributions on edges that yield the exact same frequencies in the long-run. Therefore, if the original pure finite-memory of $\mathcal{P}_1$ is surely winning, the randomized one is almost-surely winning.

In the two-player case, this approach cannot be used as frequencies are not well-defined, since the strategy of $\mathcal{P}_2$ is unknown. Consider a game which needs perfect balance between frequencies of appearance of two sets of edges in a play to be winning (Fig. 7.2). To almost-surely achieve mean-payoff vector $(0, 0)$, $\mathcal{P}_1$ must ensure that the long-term balance between edges (s_4, s_5) and (s_4, s_6) is the same as the one between edges (s_1, s_3) and (s_1, s_2). This is achievable with memory as it suffices to react immediately to compensate the choice of $\mathcal{P}_2$. However, given a randomized memoryless strategy of $\mathcal{P}_1$, $\mathcal{P}_2$ always has a strategy to enforce that the long-term frequency is unbalanced, and thus the game cannot be won almost-surely by $\mathcal{P}_1$ with such a strategy. Achieving expected mean-payoff $(0, 0)$ is also excluded.

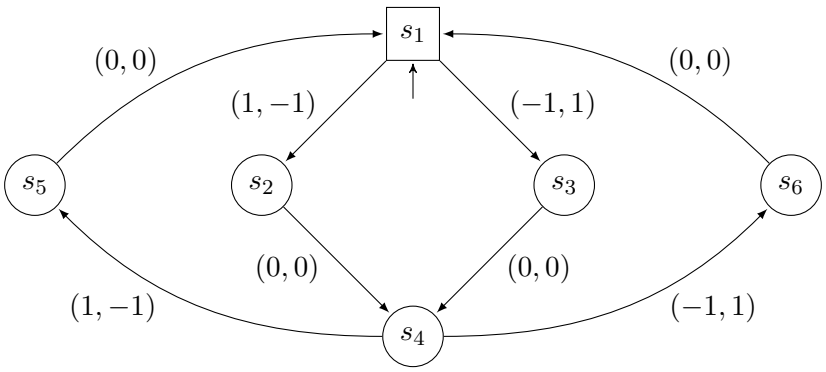


Figure 7.2: Memory is needed to enforce perfect long-term balance.

Proof. We begin with the one-player case. Let G_p be a multi mean-payoff parity game. Let $\lambda_1^{pf} \in \Lambda_1^{PF}$ be the pure finite-memory strategy of the player. Since it is pure and finite, its unique outcome is a regular play $\pi = \rho_1 \cdot (\rho_2)^\omega$, with $\rho_1 \in S^*$ and $\rho_2 \in S^+$. Let $\phi = \mathsf{MeanInf}_{G_p}(\mu) \cap \mathsf{Parity}_{G_p}$ be the multi mean-payoff parity objective for some threshold vector $\mu \in \mathbb{Q}^k$. Assume strategy λ_1^{pf} satisfies

objective ϕ. Then it also yields expected mean-payoff at least equal to μ thanks to Lemma 7.3. We claim that there exists a randomized memoryless strategy $\lambda_1^{rm} \in \Lambda_1^{RM}$ that almost-surely satisfies ϕ and ensures expectation μ, and we show how to build it.

We denote concatenation by the $\cdot$ symbol. Given a finite prefix $\rho \in S^*$ and two states $s, s' \in S$, we resp. denote by $\mathsf{occ}(s, \rho)$ and $\mathsf{occ}((s, s'), \rho)$ the number of occurences of the state s and the transition (s, s') in the prefix ρ. We add the subscript $\circ$ when we count the first state of the prefix as the successor of the last one (i.e., the prefix represents a cycle in the game graph). That is, $\mathsf{occ}_\circ(*, \rho) = \mathsf{occ}(*, \rho \cdot \mathsf{First}(\rho))$.

Let us consider the mean-payoff of the unique outcome of strategy λ_1^{pf}. Recall that for a play $\pi \in \mathsf{Plays}(G_p)$, written as the sequence $\pi = s^1, s^2, s^3 \dots$, we have $\underline{\mathsf{MP}}(\pi) = \liminf_{n\to\infty} \frac{1}{n} \sum_{1 \le i < n} w(s^i, s^{i+1})$. Since the play induced by λ_1^{pf} is regular, the limit is well-defined and we may express the mean-payoff in terms of frequencies, that is

$$\underline{\mathsf{MP}}(\pi) = \sum_{(s,s') \in E} w(s, s') \cdot \mathsf{freq}_\infty((s, s')),$$

where freq_∞ denotes the long-term frequency of a transition defined as

$$\forall\, (s, s') \in E, \quad \mathsf{freq}_\infty((s, s')) = \frac{\mathsf{occ}_\circ((s, s'), \rho_2)}{|\rho_2|}.$$

We define the randomized memoryless strategy λ_1^{rm} in the following way: $\forall\, s, s' \in S$, $(s, s') \in E$, $X = \{(s, t) \,|\, t \in S, (s, t) \in (\rho_1 \cdot \mathsf{First}(\rho_2))\}$,

$$\lambda_1^{rm}(s)(s') = \begin{cases} \dfrac{1}{|X|} \text{ if } s \in \rho_1 \wedge s \notin \rho_2, \\ \dfrac{\mathsf{occ}_\circ((s, s'), \rho_2)}{\mathsf{occ}(s, \rho_2)} \text{ if } s \in \rho_2, \\ 0 \text{ otherwise.} \end{cases}$$

Intuitively, we fix a uniform distribution over edges of the finite prefix ρ_1 as we only need to ensure reaching the *bottom strongly connected component*[1] (BSCC)

[1] Maximal subgraphs of the Markov chain that are strongly connected and without exiting

defined by ρ_2 with probability 1, and the relative frequencies in ρ_1 do not matter (because these weights and priorities are negligible in the long run). On the contrary, we use the exact frequencies for transitions of ρ_2 as they prevail long-term wise. Note that λ_1^{rm} is a correctly defined randomized memoryless strategy.

Obviously, λ_1^{rm} yields a Markov chain over states of $(\rho_1 \cup \rho_2)$ such that states of $(\rho_1 \setminus \rho_2)$ are transient and states of ρ_2 constitute a BSCC that is reached with probability one. Thus, the mean-payoff induced by λ_1^{rm} is almost-surely totally dependent on this BSCC mean-payoff value. As a consequence, proving that transition frequencies in the BSCC are exactly the same as frequencies freq_∞ defined by λ_1^{pf} will imply the claim on mean-payoff. Moreover, parity will remain satisfied almost-surely as the sets of infinitely often visited states will be the same for both the pure and the randomized strategy. Let $T = \{t_1, t_2, \ldots, t_m\}$ be the set of states that appear in ρ_2. This BSCC is an ergodic Markov chain M_e with the following matrix of transition probabilities:

$$P = \begin{array}{c} \\ t_1 \\ \vdots \\ t_m \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & \cdots & t_m \end{array} \\ \begin{pmatrix} \frac{\mathsf{occ}_\circ((t_1,t_1),\rho_2)}{\mathsf{occ}(t_1,\rho_2)} & & \\ & \ddots & \\ & & \frac{\mathsf{occ}_\circ((t_m,t_m),\rho_2)}{\mathsf{occ}(t_m,\rho_2)} \end{pmatrix} \end{array}.$$

Classical analysis of ergodic Markov chains grants the existence of a unique probability vector ν such that $\nu P = \nu$, i.e.

$$\forall\, 1 \leq i \leq m,\ \nu_i = \sum_{1 \leq j \leq m} \frac{\mathsf{occ}_\circ\left((t_j,t_i),\rho_2\right)}{\mathsf{occ}\left(t_j,\rho_2\right)} \cdot \nu_j.$$

This vector ν represents the occurence frequency of each state in an infinite run over the Markov chain. It is easy to see that the unique probability vector ν that satisfies $\nu P = \nu$ is

$$\nu = \left(\frac{\mathsf{occ}(t_1,\rho_2)}{|\rho_2|}, \quad \ldots \quad , \frac{\mathsf{occ}(t_m,\rho_2)}{|\rho_2|} \right).$$

edges of positive probability. See [GS97] or [BK08] for an introduction to the theory of Markov chains.

Moreover, given a transition of the Markov chain, its frequency is simply the product of the frequency of its starting state by the probability of the transition when the chain is in this state: for all $t, t' \in T$, we have the equality $\mathsf{freq}_\infty^{M_e}((t,t')) = \nu(t) \cdot P(t,t')$. By definition of ν and P, that is

$$\mathsf{freq}_\infty^{M_e}((t,t')) = \frac{\mathsf{occ}_\circ((t,t'),\rho_2)}{|\rho_2|} = \mathsf{freq}_\infty((t,t')),$$

thus proving that the randomized strategy λ_1^{rm} *almost-surely* yields the same mean-payoff and parity as the pure finite-memory one λ_1^{pf}. The expected value threshold is also verified by Lemma 7.3.

Now it remains to show that this does not carry over to two-player games. Indeed, we show that randomized memoryless strategies cannot replace pure finite-memory ones for the expectation semantics, even without parity. By Lemma 7.3, this implies that it cannot be verified for almost-sure semantics either. Consider the game depicted in Fig. 7.2. Player $\mathcal{P}_1$ has a pure finite-memory strategy λ_1^{pf} that ensures $\underline{\mathsf{MP}}(\pi) \geq (0,0)$, against all strategy λ_2 of $\mathcal{P}_2$. This strategy is simply to take the opposite choice of $\mathcal{P}_2$: $\lambda_1^{pf}(*s_2s_4) = s_6$ and $\lambda_1^{pf}(*s_3s_4) = s_5$. Now suppose $\mathcal{P}_1$ uses a randomized memoryless strategy λ_1^{rm} such that $\lambda_1^{rm}(s_4)(s_5) = p$ and $\lambda_1^{rm}(s_4)(s_6) = 1-p$, for some $p \in [0,1]$. We claim that whatever the value of p, there exists a counter-strategy λ_2 for $\mathcal{P}_2$ such that $\mathbb{E}_{s_1}^{G_p[\lambda_1^{rm},\lambda_2]}(\underline{\mathsf{MP}}) \not\geq (0,0)$. Suppose $p \geq 1/2$ and let $\lambda_2(s_1) = s_2$. Then, we have

$$\begin{aligned}\mathbb{E}_{s_1}^{G_p[\lambda_1^{rm},\lambda_2]}(\underline{\mathsf{MP}}) &= \frac{(1,-1) + [p \cdot (1,-1) + (1-p) \cdot (-1,1)]}{4} \\ &= \frac{1}{2}(p,-p) \not\geq (0,0).\end{aligned}$$

Now suppose $p < 1/2$ and let $\lambda_2(s_1) = s_3$. Then, we have

$$\begin{aligned}\mathbb{E}_{s_1}^{G_p[\lambda_1^{rm},\lambda_2]}(\underline{\mathsf{MP}}) &= \frac{(-1,1) + [p \cdot (1,-1) + (1-p) \cdot (-1,1)]}{4} \\ &= \frac{1}{2}(p-1,1-p) \not\geq (0,0).\end{aligned}$$

This shows that memory is needed to achieve expectation $(0,0)$ and concludes our proof. □

7.4 Single Mean-Payoff Parity Games

Randomized memoryless strategies can replace pure finite-memory ones for single mean-payoff parity games. The proof is in two steps.

First, we show that it is the case for the simpler case of *mean-payoff Büchi games* (Lemma 7.7). Suppose $\mathcal{P}_1$ has a pure finite-memory winning strategy for such a game. We use the existence of particular pure memoryless strategies on winning states: the classical attractor for Büchi states, and a strategy that ensures that cycles of the outcome have positive energy (whose existence follows from [CD12]). We build an almost-surely randomized memoryless winning strategy for $\mathcal{P}_1$ by mixing those strategies in the probability distributions, with sufficient probability over the strategy that is good for energy.

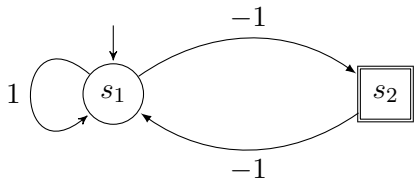


Figure 7.3: Mixing strategies that are resp. *good for Büchi* and *good for energy*.

We illustrate this construction on the simple game G depicted in Fig. 7.3. Let $\lambda_1^{pf} \in \Lambda_1^{PF}$ be a strategy of $\mathcal{P}_1$ such that $\mathcal{P}_1$ plays (s_1, s_1) for 8 times, then plays (s_1, s_2) once, and so on. This strategy ensures surely winning for the objective $\phi = \mathsf{MeanInf}_G(3/5) \cap \mathsf{Buchi}_G(\{s_2\})$. Obviously, $\mathcal{P}_1$ has a pure memoryless strategy that ensures winning for the Büchi objective: playing (s_1, s_2). On the other hand, he also has a pure memoryless strategy that ensures cycles of positive energy: playing loop (s_1, s_1). Let $\lambda_1^{rm} \in \Lambda_1^{RM}$ be the strategy defined as follows: play edge (s_1, s_2) with probability γ and edge (s_1, s_1) with the remaining probability. This strategy is almost-surely winning for ϕ for sufficiently small values of γ (e.g., $\gamma = 1/9$).

Second, we extend this result to *mean-payoff parity games* using an induction on the number of priorities and the size of games (Lemma 7.10). We consider *subgames* that reduce to the MP Büchi and MP coBüchi cases. For MP coBüchi games, pure memoryless strategies are known to suffice [CHJ05].

7.4.1 Büchi case

A simpler case of the parity objective is the Büchi objective (see Sect. 2.3). It corresponds to parity with priorities $\{0, 1\}$. Given a game $G = (S_1, S_2, E, w)$, let $T \subseteq S$ denote the set of Büchi states such that a play is winning if it visits infinitely often states of the set T.

We first state results on these Büchi objectives, as they are conceptually simpler to understand. Proof arguments for parity are more involved and make use of results on Büchi objectives.

Recall the notion of ε-optimality introduced in Sect. 2.2.4. In Fig. 7.4, we present a game where finite-memory optimal strategies do not exist for $\mathcal{P}_1$, whereas finite-memory ε-optimal strategies exist for any $\varepsilon > 0$. Indeed, $\mathcal{P}_1$ has to visit s_2 infinitely often, but he has to delay its visits of s_2 for longer and longer intervals in order to achieve value 1 for the mean-payoff. This requires infinite memory. However, for any fixed $\varepsilon > 0$, $\mathcal{P}_1$ can set up a finite-memory strategy that visits s_2 rarely enough to achieve mean-payoff $1 - \varepsilon$.

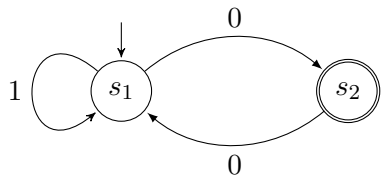


Figure 7.4: Mean-payoff Büchi requires infinite memory for optimality.

We show that finite-memory strategies can be traded for randomized memoryless ones in mean-payoff Büchi games. Precisely, we prove that ε-optimality can as well be achieved by randomized memoryless strategies (when relaxing to almost-sure semantics). We first need to state two useful lemmas granting the existence of pure memoryless strategies that are resp. *good-for-energy* or *good-for-Büchi* strategies, in all states that are winning for the mean-payoff Büchi objective. These strategies will help us build the needed ε-optimal strategies.

Lemma 7.5 (Extension of [CD12, Lemma 4]). *Let $G = (S_1, S_2, E, w)$ be a game with $T \subseteq S$ the set of Büchi states. Let $\mathsf{Win} \subseteq S$ be the set of winning states for the mean-payoff Büchi objective with worst-case mean-payoff threshold zero. For all $s \in \mathsf{Win}$, $\mathcal{P}_1$ has a uniform (i.e., independent of the starting state)*

memoryless good-for-energy strategy λ_1^{gfe} whose outcome never leaves the set Win, such that any cycle of this outcome has a non-negative energy level.

Lemma 7.6 (Classical attractor strategy). *Let $G = (S_1, S_2, E, w)$ be a game with $T \subseteq S$ the set of Büchi states. Let $\mathsf{Win} \subseteq S$ be the set of winning states for the mean-payoff Büchi objective with worst-case mean-payoff threshold zero. For all $s \in \mathsf{Win}$, $\mathcal{P}_1$ has a uniform (i.e., independent of the starting state) memoryless good-for-Büchi strategy $\lambda_1^{\Diamond T}$, an attractor strategy for T, whose outcome never leaves the set Win, such that it ensures reaching T in at most $|S|$ steps.*

The randomized memoryless strategy of $\mathcal{P}_1$ will thus consist in mixing these two strategies, with a very low probability on the good-for-Büchi strategy. Indeed, the Büchi objective will be almost-surely satisfied whatever this probability is, provided it is strictly positive. On the other hand, by giving more weight to the good-for-energy strategy, $\mathcal{P}_1$ can obtain a mean-payoff that is arbitrary close to the optimum.

Lemma 7.7. *In mean-payoff Büchi games, ε-optimality can be achieved surely by pure finite-memory strategies and almost-surely by randomized memoryless strategies.*

Proof. Let $G = (S_1, S_2, E, w)$ be a game with $T \subseteq S$ the set of Büchi states and $s_{\mathsf{init}} \in S$ the initial state. We consider the mean-payoff objective with worst-case threshold zero (w.l.o.g.). Let $\mathsf{Win} \subseteq S$ be the set of winning states for the mean-payoff Büchi objective. By Lemma 7.5 and Lemma 7.6, for all $s \in \mathsf{Win}$, $\mathcal{P}_1$ has two uniform memoryless strategies λ_1^{gfe} and $\lambda_1^{\Diamond T}$, whose outcomes never leave the set Win, such that λ_1^{gfe} ensures that any cycle of its outcome has non-negative energy, and $\lambda_1^{\Diamond T}$, an attractor strategy for T, ensures reaching T in at most $|S|$ steps.

We first build ε-optimal *pure finite-memory* strategies based on these two pure memoryless strategies. Let $\varepsilon > 0$. As usual, W denotes the largest absolute weight on any edge. Let us define $\lambda_1^{pf} \in \Lambda_1^{PF}$ such that (a) it plays λ_1^{gfe} for $\left\lceil \frac{2 \cdot W \cdot |S|}{\varepsilon} \right\rceil - |S|$ steps, then (b) it plays $\lambda_1^{\Diamond T}$ for $|S|$ steps, then again (a). This ensures that T is visited infinitely often as $\lambda_1^{\Diamond T}$ is played infinitely many times for $|S|$ steps in a row. Furthermore, the total cost of phases (a) + (b) is bounded by $-2 \cdot W \cdot |S|$, and thus the mean-payoff of the outcome is at least $-\varepsilon$, against any strategy of the adversary.

Second, we show that based on the same pure memoryless strategies, it is possible to obtain almost-surely ε-optimal *randomized memoryless* strategies. That is,

$$\forall\, \varepsilon > 0,\ \exists\, \lambda_1^{rm} \in \Lambda_1^{RM},\ \forall\, \lambda_2 \in \Lambda_2,$$
$$\mathbb{P}_{s_{\mathsf{init}}}^{G[\lambda_1^{rm},\lambda_2]}(\pi \vDash \square\diamondsuit T) = 1 \ \wedge\ \mathbb{P}_{s_{\mathsf{init}}}^{G[\lambda_1^{rm},\lambda_2]}(\underline{\mathsf{MP}}(\pi) \geq -\varepsilon) = 1.$$

Note that pure memoryless strategies suffice for $\mathcal{P}_2$ as he essentially has to win against the Büchi *or* the mean-payoff criterion [BMOU11]. Therefore, given $\varepsilon > 0$, we need to build some strategy $\lambda_1^{rm} \in \Lambda_1^{RM}$ such that

$$\forall\, \lambda_2^{pm} \in \Lambda_2^{PM},\ \mathbb{P}_{s_{\mathsf{init}}}^{G[\lambda_1^{rm},\lambda_2^{pm}]}(\pi \vDash \square\diamondsuit T) = 1 \ \wedge\ \mathbb{P}_{s_{\mathsf{init}}}^{G[\lambda_1^{rm},\lambda_2^{pm}]}(\underline{\mathsf{MP}}(\pi) \geq -\varepsilon) = 1.$$

We build such a strategy as follows:

$$\forall s \in S,\ \lambda_1^{rm}(s) = \begin{cases} \lambda_1^{gfe}(s) \text{ with probability } 1-\gamma, \\ \lambda_1^{\diamondsuit T}(s) \text{ with probability } \gamma, \end{cases}$$

for some *well-chosen* $\gamma \in]0,1[$.

It is straightforward to see that the Büchi objective is almost-surely satisfied for all values of $\gamma > 0$ as at all times, the probability of playing according to $\lambda_1^{\diamondsuit T}$ for $|S|$ steps in a row, and thus ensuring a visit of T, is $\gamma^{|S|}$, which is strictly positive.

It remains to study if it is always possible to choose such a constant γ such that objective $\mathsf{MeanInf}_G(-\varepsilon)$ is almost-surely satisfied. Consider such a strategy $\lambda_1^{rm} \in \Lambda_1^{RM}$ and some fixed strategy $\lambda_2^{pm} \in \Lambda_2^{PM}$ of $\mathcal{P}_2$: the game reduces to the finite Markov chain $M = G[\lambda_1^{rm}, \lambda_2^{pm}]$.

Suppose λ_2^{pm} is winning for $\mathcal{P}_2$. Thus, $\mathbb{P}_{s_{\mathsf{init}}}^{M}(\underline{\mathsf{MP}}(\pi) < -\varepsilon) > 0$. The mean-payoff depends on limit behavior: the probability measure of plays that do not enter in a bottom strongly connected component (BSCC) is zero [BK08], whereas in a BSCC, the expected mean-payoff is the same in all states and it is obtained almost-surely.[2] This implies that there exists some BSCC $\mathcal{B}$ in M such that $\mathbb{P}_{s_{\mathsf{init}}}^{M}(\diamondsuit\mathcal{B}) > 0$ and $\mathbb{E}^{\mathcal{B}}(\underline{\mathsf{MP}}) < -\varepsilon$.

[2] This follows from definition of BSCCs and prefix-independence of the mean-payoff. Further discussion of the subject is presented in Part IV.

We claim that it is possible to choose γ such that all BSCCs, in all Markov chains induced by pure memoryless strategies of $\mathcal{P}_2$, have expectation greater than or equal to ε, thus proving that strategy λ_1^{rm} is almost-surely ε-optimal with regard to the mean-payoff value function. Intuitively, the smaller this constant γ is chosen, the nearer will the expected mean-payoff induced by λ_1^{rm} be to the one induced by λ_1^{gfe}, that is at least zero. Since the number of pure memoryless strategies of $\mathcal{P}_2$ is finite, and so is the number of BSCCs induced by λ_1^{rm} (regardless of the exact value of $\gamma \in]0,1[$, we obtain the same BSCCs in terms of states and edges), one can compute a suitable γ for each of them, and take the mininum to ensure that the property will be satisfied in all possible cases.

Therefore, let us fix some strategy λ_2^{pm} of $\mathcal{P}_2$, and some BSCC $\mathcal{B}$ of the induced Markov chain when played against strategy λ_1^{rm} of $\mathcal{P}_1$. It remains to show that (*claim*) there exists $\gamma \in]0,1]$ such that $\mathbb{E}^{\mathcal{B}(\gamma)}(\underline{\mathsf{MP}}) \geq -\varepsilon$ to conclude this proof. Observe that we write $\mathcal{B}(\gamma)$ as transition probabilities inside $\mathcal{B}$ depend on γ.

By contradiction, suppose the claim is false. Precisely, we assume that (*contradiction hypothesis*) for all $\gamma \in]0,1]$, we have that $\mathbb{E}^{\mathcal{B}(\gamma)}(\underline{\mathsf{MP}}) < -\varepsilon$.

Besides, observe that for $\gamma = 0$, strategy λ_1^{rm} is exactly equal to λ_1^{gfe}. As we know that λ_1^{gfe} ensures a worst-case mean-payoff at least equal to zero, we trivially deduce that $\mathbb{E}^{\mathcal{B}(0)} \geq 0$. This implies that

$$\sup_{\gamma \in [0,1]} \mathbb{E}^{\mathcal{B}(\gamma)} \geq \mathbb{E}^{\mathcal{B}(0)} \geq 0.$$

Notice that in this case, interval $[0,1]$ is closed.

By results in the literature, it is known that this supremum is continuous. See for example Solan [Sol03] on the continuity of the optimal expected value function in the general context of competitive Markov decision processes (equivalent to $2\frac{1}{2}$-player games). Therefore, we have that

$$\sup_{\gamma \in]0,1]} \mathbb{E}^{\mathcal{B}(\gamma)} = \sup_{\gamma \in [0,1]} \mathbb{E}^{\mathcal{B}(\gamma)} \geq \mathbb{E}^{\mathcal{B}(0)} \geq 0.$$

On the other hand, by (*contradiction hypothesis*), we also have that

$$\sup_{\gamma \in]0,1]} \mathbb{E}^{\mathcal{B}(\gamma)} \leq -\varepsilon.$$

Since ε is strictly positive, there is a clear contradiction, which concludes our proof. □

7.4.2 Parity Case

Given the results for mean-payoff Büchi games, we now consider the more general case of mean-payoff parity games. Given a game $G_p = (S_1, S_2, E, w, p)$ and a set $A \subseteq S$, recall that $G_p \restriction A$ denotes the subgame induced by A, as defined in Sect. 2.1. Also, recall the notion of attractors, again defined in Sect. 2.1. For any set $A \subseteq S$, $\mathsf{Attr}^{\mathcal{P}_i}_{G_p}(A)$ contains all the states in S from which $\mathcal{P}_i$ can force a visit to A, and it is well known that $\mathcal{P}_1$ has a pure memoryless strategy to force such a visit from those states. Also, it is clear that $\mathcal{P}_i$ does not have a strategy to leave the states in $S \setminus \mathsf{Attr}^{\mathcal{P}_i}_{G_p}(A)$. As direct consequence, we have the following proposition.

Proposition 7.8. *Let $G_p = (S_1, S_2, E, w, p)$ be a game and $s_{\mathsf{init}} \in S$ an initial state. Let $U \subseteq S$ and $\mathsf{Attr}^{\mathcal{P}_1}_{G_p}(U)$ be such that $B = S \setminus \mathsf{Attr}^{\mathcal{P}_1}_{G_p}(U)$ is non-empty, then $G_p \restriction B$ is a deadlock-free subgame.*

The following lemma states that optimal pure memoryless strategies exist for $\mathcal{P}_1$ in games with mean-payoff coBüchi objectives (i.e., parity with priorities taken in $\{1, 2\}$). For mean-payoff Büchi objectives, we showed in Lemma 7.7 that, for all $\varepsilon > 0$, ε-optimal randomized memoryless strategies exist.

Lemma 7.9 ([CHJ05, Theorem 5]). *Let $G_p = (S_1, S_2, E, w, p)$ be a game with priorities $\{1, 2\}$, and $\textsc{Win}^p_{\geq 0}$ be the set of nodes in G_p from which $\mathcal{P}_1$ wins the mean-payoff coBüchi objective for threshold zero (w.l.o.g.). Then from all states in $\textsc{Win}^p_{\geq 0}$, $\mathcal{P}_1$ has a pure memoryless winning strategy for the coBüchi mean-payoff objective for threshold zero.*

We now establish that ε-optimal randomized memoryless strategies also exist for mean-payoff *parity* games, and thus, can replace pure finite-memory ones.

Lemma 7.10. *Let $G_p = (S_1, S_2, E, w, p)$ and $\textsc{Win}^p_{\geq 0}$ be the set of nodes in G_p from which $\mathcal{P}_1$ wins the mean-payoff parity objective for threshold zero. Then for all $\varepsilon > 0$, there exists $\lambda_1^{rm} \in \Lambda_1^{RM}$, such that for all $s \in \textsc{Win}^p_{\geq 0}$ and for all $\lambda_2 \in \Lambda_2$, we have that:*

$$\mathbb{P}_s^{G[\lambda_1^{rm}, \lambda_2]}(\underline{\mathsf{MP}}(\pi) \geq -\varepsilon) = 1 \ \wedge \ \mathbb{P}_s^{G[\lambda_1^{rm}, \lambda_2]}(\mathsf{Par}(\pi) \bmod 2 = 0) = 1.$$

Proof. The proof is by induction on the lexicographic order $\preceq$ on games, defined as follows: $G_p^1 \preceq G_p^2$ if G_p^1 has less priorities than in G_p^2 or G_p^1 has the same priorities than in G_p^2 but less states. Clearly, this lexicographic order is well-founded.

The base cases are twofold: one for the number of states, and one for priorities. First, if the game is such that $|S| = 1$, then obviously, if $\mathcal{P}_1$ can win, he can do so with a pure memoryless strategy, which respects the claim. Second, for two priorities. W.l.o.g., we can assume that all priorities are either in $\{0, 1\}$ or in $\{1, 2\}$. Those cases resp. correspond to mean-payoff Büchi and mean-payoff coBüchi games. The result for mean-payoff Büchi games has been established in Lemma 7.7, while the result for mean-payoff coBüchi games is a direct consequence of Lemma 7.9, as pure memoryless strategies are a special case of randomized memoryless strategies.

Let us now consider the inductive case. Suppose we have a mean-payoff parity game G_p with m priorities and $|S|$ states. W.l.o.g., we can make the assumption that the lowest priority in G_p is either 0 or 1, otherwise we subtract an even number to all priorities so that we are in that case. We introduce the following sets: $U_0 = \{s \in \textsc{Win}^p_{\geq 0} \mid p(s) = 0\}$ and $U_1 = \{s \in \textsc{Win}^p_{\geq 0} \mid p(s) = 1\}$.

We consider the two possible following situations: (A) U_0 is empty or (B) it is not.

Case (A). First assume U_0 *is empty*. In that case U_1 is not empty. Let us consider $A_2 = \mathsf{Attr}^{\mathcal{P}_2}_{G_p}(U_1)$ the attractor of $\mathcal{P}_2$ for U_1. It must be the case that $\textsc{Win}^p_{\geq 0} \setminus A_2$ is non-empty, otherwise this would contradict the fact that $\mathcal{P}_1$ is winning the parity objective from states in $\textsc{Win}^p_{\geq 0}$. Indeed, if it was not the case, then $\mathcal{P}_2$ would be able to force an infinite number of visits to U_1 from all states in $\textsc{Win}^p_{\geq 0}$, and the parity would be odd as U_0 is empty, a contradiction with the definition of $\textsc{Win}^p_{\geq 0}$.

(i) Let $B = \textsc{Win}^p_{\geq 0} \setminus A_2$. First note that, as B is non-empty, by Proposition 7.8, $G_p \restriction B$ is a deadlock-free subgame. Also, note that from all states in B, it must be the case that $\mathcal{P}_1$ has a winning strategy that does not require visits of the states outside B, i.e., states in A_2, for otherwise this would lead to a contradiction with the fact that $\mathcal{P}_1$ is winning the parity objective in $\textsc{Win}^p_{\geq 0}$. So all states in the subgame $G_p \restriction B$ are winning for $\mathcal{P}_1$. The game $G_p \restriction B$ does not contain states with priority 0, and so we can apply our induction hypothesis

to conclude that $\mathcal{P}_1$ has a memoryless randomized strategy from all states in B, as $(G_p \restriction B) \preceq G_p$ since it has one less priority.

(ii) Now, let us concentrate on states in A_2. Let $A_1 = \mathsf{Attr}^{\mathcal{P}_1}_{G_p}(B)$. From states in A_1, $\mathcal{P}_1$ has a pure memoryless strategy to reach states in B, and so from there $\mathcal{P}_1$ can play as in $G_p \restriction B$, and we are done. Let $C = A_2 \setminus A_1$. If C is empty, we are done. Otherwise, by Proposition 7.8, $G_p \restriction C$ is a deadlock-free subgame ($\mathcal{P}_2$ can force to stay within C). We conclude that all states in this game must be winning for $\mathcal{P}_1$. This game has the same minimal priority than in the original game (i.e., priority 1) but it has at least one state less, and so we can apply our induction hypothesis to conclude that $\mathcal{P}_1$ has a memoryless randomized strategy from all states in C.

Therefore, by (i) and (ii), $\mathcal{P}_1$ has a memoryless randomized strategy from all states in $\textsc{Win}^p_{\geq 0}$, which proves the claim in that case.

Case (B). Second, assume U_0 *is not empty*. Let us consider the attractor set $A_1 = \mathsf{Attr}^{\mathcal{P}_1}_{G_p}(U_0)$.

(iii) First, consider the case where $A_1 = \textsc{Win}^p_{\geq 0}$. In this case, it means that $\mathcal{P}_1$ can force a visit to states in U_0 from any states in $\textsc{Win}^p_{\geq 0}$. So, we conclude that $\mathcal{P}_1$ wins in G_p the mean-payoff Büchi game with threshold 0, and by Lemma 7.7, we conclude that $\mathcal{P}_1$ has a memoryless randomized strategy from all states in G_p for almost-surely winning the parity game with mean-payoff threshold 0 so we are done.

(iv) Second, consider the case where $B = \textsc{Win}^p_{\geq 0} \setminus A_1$ is non-empty. Then by Proposition 7.8, $G_p \restriction B$ is a deadlock-free subgame. So $\mathcal{P}_2$ can force to stay within B in the original game and so we conclude that all states in the game $G_p \restriction B$ are winning for $\mathcal{P}_1$. As $G_p \restriction B$ does not contain states of priority 0, and thus has at least one less priority, we can apply the induction hypothesis to conclude that $\mathcal{P}_1$ has a memoryless randomized strategy from all states in B.

Therefore, by (iii) and (iv), $\mathcal{P}_1$ has a memoryless randomized strategy from all states in $\textsc{Win}^p_{\geq 0}$, which also proves the case.

As we have established the claim in both possible cases, this concludes the proof. □

7.5 Wrap-up

We sum up results for these different classes of games in Theorem 7.11 (also see Table 7.1).

Theorem 7.11 (Trading finite memory for randomness). *The following assertions hold:*

(1) Randomized strategies are exactly as powerful as pure strategies for energy objectives. Randomized memoryless strategies are not as powerful as pure finite-memory strategies for almost-sure winning in one-player and two-player energy, multi energy, energy parity and multi energy parity games.

(2) Randomized memoryless strategies are not as powerful as pure finite-memory strategies for almost-sure winning in two-player multi mean-payoff games.

(3) In one-player multi mean-payoff parity games, and two-player single mean-payoff parity games, if there exists a pure finite-memory sure winning strategy, then there exists a randomized memoryless almost-sure winning strategy.

Proof. (1) For energy games, results follow from Lemma 7.2. (2) For two-player multi mean-payoff games, they follow from Lemma 7.4. (3) For one-player multi mean-payoff games, they follow from Lemma 7.4. For two-player single mean-payoff parity, they are direct consequence of Lemma 7.10. □

We close this section by observing that there are even more powerful classes of strategies. Their study, as well as their practical interest, remains open.

Lemma 7.12. *Randomized finite-memory strategies are strictly more powerful than both randomized memoryless and pure finite-memory strategies for multi-mean payoff games with expectation semantics, even in the one-player case.*

The intuition is essentially that memory permits to achieve an exact payoff by sticking to a given side, while randomization permits to combine payoffs of pure strategies to achieve any linear combination in between.

Proof. Consider the game G depicted in Fig. 7.5. Whatever the pure finite-memory strategy of $\mathcal{P}_1$, the only achievable mean-payoff values are $(1, -1)$ (if (s_1, s_2) is never taken) and $(-1, 1)$ (if (s_1, s_2) is taken). This is also true for

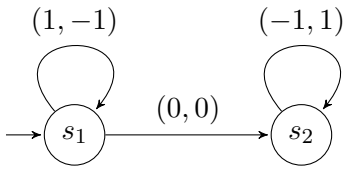


Figure 7.5: Randomized finite memory is strictly more powerful than randomized memorylessness and pure finite memory.

randomized memoryless strategies: either the probability of (s_1, s_2) is null and the mean-payoff has value $(1, -1)$, or this probability is strictly positive, and the mean-payoff almost-surely has value $(-1, 1)$ as the probability mass will eventually reach s_2. In contrast, expected value $(0, 0)$ is achievable by a randomized finite-memory strategy. Indeed, consider the strategy that tosses a coin in its first visit of s_1 to decide if it will always play edge (s_1, s_1) or if it will play edge (s_1, s_2) once and then always play edge (s_2, s_2). This strategy only needs one bit of memory and one bit to encode probabilities, and still, it is strictly more powerful than any amount of pure memory or any arbitrary high precision for probabilities without memory. □

Part III

Window Objectives

CHAPTER 8

Window Mean-Payoff

Introduction ◇ Definition ◇ Relation with Classical Objectives

Applicability of total-payoff and mean-payoff objectives can prove to be challenging for synthesis tasks for at least two reasons.

First, they characterize *long-run* behaviors over infinite plays. A desired property in many practical problems is to provide bounds on *time frames* in which an acceptable behavior can be witnessed.

Second, both objectives face *complexity barriers*. Whether one-dimension mean-payoff and total-payoff games belong to P or not is a long-standing open problem. Moreover, we proved in Theorem 4.8 that total-payoff becomes undecidable in multi-dimension games.

We introduce *conservative approximations* of these objectives, based on the *window mean-payoff objective* (WMP) that considers the payoff over a local finite window sliding along a play, instead of the whole play. This objective strengthen classical quantitative specifications with timing guarantees. Moreover, we prove in the following chapters that it benefits from improved tractability.

Window objectives were studied in joint work with Chatterjee, Doyen and Raskin [CDRR13a, CDRR13b].

8.1 Introduction

Window objectives. We propose to study variants of the classical total-payoff and mean-payoff objectives that can express guarantees on time frames. Namely, the *bounded window mean-payoff*[1] and *fixed window mean-payoff* objectives.

In a window mean-payoff objective, instead of the long-run average along the whole play, we consider payoffs over a local bounded window sliding along the play. The objective is that the average weight must be at least zero before the end of every bounded window from some point on. This can be seen as a strengthening of the mean-payoff objective (resp. of the total-payoff objective if we require that the window objective is satisfied from the beginning of the play rather than from some point on). That is, winning for the window mean-payoff implies winning for the mean-payoff.

In the fixed window variant, the bound on the window size is a parameter. In the bounded window, the question is whether there exists a finite bound such that the corresponding fixed window objective is satisfied on the considered play.

Attractive features for window objectives. First, they are a strengthening of total-payoff and mean-payoff objectives and hence provide *conservative approximations* (see Lemma 8.3 for a formal definiton).

Second, the window variant is very natural to study in system analysis. Mean-payoff objectives require the average payoff to satisfy certain threshold in the long-run (or in the limit of the infinite path), whereas the window objectives require to provide guarantee on the average, not in the limit, but within a bounded time, and thus provide *better time guarantee* than the mean-payoff objectives.

Third, the window parameter provides *flexibility*: it can be adjusted specific to applications requirement of strong or weak time guarantee.

Finally, our variant in the single dimension is more *computationally tractable*, which makes it an attractive alternative to classical objectives.

Structure. This chapter presents window objectives and summarizes the main results. We then divide our analysis into two chapters: Chap. 9 for one-dimension games and Chap. 10 for multi-dimension games. Core literature on classical mean-payoff and total-payoff objectives is presented in Sect. 2.3.2.

[1] We name our objective by analogy with the mean-payoff, but equivalent definition could be used based on the total-payoff.

8.1.1 Assumptions

Pure strategies. We would like to stress on a restriction present in Part III with regard to the general setting described in Chap. 2. Throughout our whole study of games with window objectives, **we only consider *pure* strategies**, for both players. Consequently, we only use *sure winning semantics*. This is the traditional setting of two-player games, and a first important model to investigate when introducing new winning objectives.

Extension to the stochastic setting. Study of window objectives in a stochastic context (as a first step, MDPs) could be a promising future work. Almost-sure semantics would adequately fit the problem.

However, it should be noticed that defining relevant problems based on expected value semantics seems challenging in the context of window objectives. Indeed, there is no global value function to optimize over plays, and considering the expectation over windows may not be the most natural choice: windows slide over individual plays while expectation describes a property linked to the tree unfolding of the MDP, i.e., sets of plays.

Hence, the first effort in extending the problem to the stochastic context should be put in adequate formalization of the interesting questions to solve.

8.1.2 Overview of Results

One-dimension games. Corresponding results are summarized in Table 8.1. We present an algorithm for the fixed window objective that is polynomial in the size of the game graph times the length of the binary encoding of weights times the size of the fixed window. Thus if the window size is polynomial, we have a *polynomial-time* algorithm.

For the bounded window problem we show that the decision problem is in NP $\cap$ coNP, and at least as hard as solving mean-payoff games. However, winning for mean-payoff games does not imply winning for the bounded window mean-payoff objective, i.e., the winning sets for mean-payoff games and bounded window mean-payoff games do not coincide. Moreover, the structure of winning strategies is also very different, e.g., in mean-payoff games both players have memoryless winning strategies, but in bounded window mean-payoff games we show that $\mathcal{P}_2$ requires infinite memory.

	one-dimension		
	complexity	$\mathcal{P}_1$ mem.	$\mathcal{P}_2$ mem.
$\underline{\mathsf{MP}}$ / $\overline{\mathsf{MP}}$	NP ∩ coNP	memoryless	
$\underline{\mathsf{TP}}$ / $\overline{\mathsf{TP}}$			
WMP: fixed polynomial window	**P-c.** (Thm. 9.6)	**memory required** $\leq$ **linear**$(\lvert S\rvert \cdot l_{\max})$ (Thm. 9.6)	
WMP: fixed arbitrary window	**P**$(\lvert S\rvert, V, l_{\max})$ (Thm. 9.6)		
WMP: bounded window problem	**NP ∩ coNP** (Thm. 9.16)	**memoryless** (Thm. 9.16)	**infinite** (Thm. 9.16)

Table 8.1: Complexity of deciding the winner and memory required in *one-dimension window games*. We denote by $l_{\max}$ the window size and by V the length of the binary encoding of weights. New results in bold (c. for complete).

	k-dimension		
	complexity	$\mathcal{P}_1$ mem.	$\mathcal{P}_2$ mem.
$\underline{\mathsf{MP}}$ / $\overline{\mathsf{MP}}$	coNP-c./NP ∩ coNP	infinite	memoryless
$\underline{\mathsf{TP}}$ / $\overline{\mathsf{TP}}$	**undec.** (Thm. 4.8)	-	-
WMP: fixed polynomial window	**PSPACE-h./EXP-e.** (Thm. 10.9)	**exponential** (Thm. 10.9)	
WMP: fixed arbitrary window	**EXP-c.** (Thm. 10.9)		
WMP: bounded window problem	**NPR-h.** (Thm. 10.10)	-	-

Table 8.2: Complexity of deciding the winner and memory required in *multi-dimension window games*. New results in bold (h. for hard, e. for easy, c. for complete, EXP for EXPTIME and NPR for non-primitive recursive).

We also show that if $\mathcal{P}_1$ wins the bounded window mean-payoff objective, then a window of size $(|S| - 1) \cdot (|S| \cdot W + 1)$ is sufficient where S is the state space, and W is the largest absolute weight value.

Finally, we show that (i) a winning strategy for the bounded window mean-payoff objective of threshold zero ensures that the mean-payoff is non-negative regardless of the strategy of the opponent; and (ii) a strategy that ensures that the mean-payoff is strictly positive is winning for the bounded window mean-payoff objective of threshold zero.

Multi-dimension games. Corresponding results are summarized in Table 8.2. Fixed window games are EXPTIME-complete (both for arbitrary dimensions with weights in $\{-1, 0, 1\}$ and for two dimensions with arbitrary weights); and if the window size is polynomial, then the problem is PSPACE-hard.

For the bounded window, the problem is non-primitive-recursive-hard (i.e., there is no primitive recursive algorithm to decide the problem).

Memory requirements. For all the problems for which we prove decidability we also characterize the memory required by winning strategies.

8.2 Definition

In window objectives, the intuition is that local deviations from the threshold must be compensated in a parameterized number of steps. We consider a *window*, sliding along a play, within which the compensation must happen. Our approach can be applied both to mean-payoff and total-payoff objectives. Since we consider *finite* windows, both versions coincide for threshold zero. Hence we present our results for mean-payoff.

8.2.1 Objectives and Decision Problems

Given a (possibly multi-dimension) two-player game $G = (S_1, S_2, E, k, w)$ and a rational threshold $v \in \mathbb{Q}^k$, we define the following objectives.[2]

[2]For brevity, we omit that $\pi \in \mathsf{Plays}(G)$.

- Given $l_{\max} \in \mathbb{N}_0$, the *good window* objective

$$\mathsf{GW}_G(v, l_{\max}) = \Big\{\pi \mid \forall\, t,\, 1 \leq t \leq k,\, \exists\, l \leq l_{\max}, \frac{1}{l}\sum_{p=0}^{l-1} w\Big(e_\pi(p, p+1)\Big)(t) \geq v(t)\Big\}, \tag{8.1}$$

  where $e_\pi(p, p{+}1)$ is the edge $(\mathsf{Last}(\pi(p)), \mathsf{Last}(\pi(p{+}1)))$, requires that for all dimensions, there exists a window starting in the first position and bounded by $l_{\max}$ over which the mean-payoff is at least equal to the threshold.

- Given $l_{\max} \in \mathbb{N}_0$, the *direct fixed window mean-payoff* objective

$$\mathsf{DirFixWMP}_G(v, l_{\max}) = \Big\{\pi \mid \forall\, j \geq 0,\ \pi(j, \infty) \in \mathsf{GW}_G(v, l_{\max})\Big\} \tag{8.2}$$

  requires that good windows bounded by $l_{\max}$ exist in all positions along the play.

- The *direct bounded window mean-payoff* objective

$$\mathsf{DirBndWMP}_G(v) = \Big\{\pi \mid \exists\, l_{\max} > 0,\ \pi \in \mathsf{DirFixWMP}_G(v, l_{\max})\Big\} \tag{8.3}$$

  asks that there exists a bound $l_{\max}$ such that the play satisfies the direct fixed objective.

- Given $l_{\max} \in \mathbb{N}_0$, the *fixed window mean-payoff* objective

$$\mathsf{FixWMP}_G(v, l_{\max}) = \Big\{\pi \mid \exists\, i \geq 0,\ \pi(i, \infty) \in \mathsf{DirFixWMP}_G(v, l_{\max})\Big\} \tag{8.4}$$

  is the *prefix-independent* version of the direct fixed window objective: it asks for the existence of a suffix of the play satisfying it.

- The *bounded window mean-payoff* objective

$$\mathsf{BndWMP}_G(v) = \Big\{\pi \mid \exists\, l_{\max} > 0,\ \pi \in \mathsf{FixWMP}_G(v, l_{\max})\Big\} \tag{8.5}$$

  is the *prefix-independent* version of the direct bounded window objective.

For any $v \in \mathbb{Q}^k$ and $l_{\max} \in \mathbb{N}_0$, the following inclusions are true:

$$\mathsf{DirFixWMP}_G(v, l_{\max}) \subseteq \mathsf{FixWMP}_G(v, l_{\max}) \subseteq \mathsf{BndWMP}_G(v), \tag{8.6}$$

$$\mathsf{DirFixWMP}_G(v, l_{\max}) \subseteq \mathsf{DirBndWMP}_G(v) \subseteq \mathsf{BndWMP}_G(v). \tag{8.7}$$

All objectives can be equivalently expressed for threshold $v = \{0\}^k$ by modifying the weight function. Hence, given any variant of the objective, the associated *decision problem* is to decide the existence of a winning strategy for $\mathcal{P}_1$ for threshold $\{0\}^k$. For complexity purposes, we distinguish *polynomial* (in the size of the game) from *arbitrary* (i.e., non-polynomial) window sizes.

Notice that all those objectives define Borel sets. Hence they are determined by Martin's theorems [Mar75, Mar98].

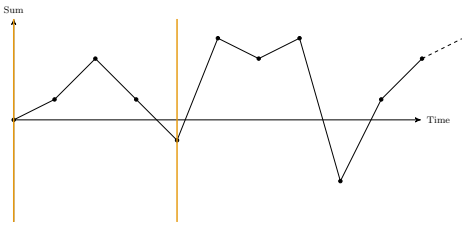
Discussion. Introduction of new objectives is a delicate task: our formalization is the result of a long modeling process. On the applicability standpoint, we are driven by potential needs in applications and the issue of timing guarantees in specifications. On the theoretical one, we pay attention in providing a model with robust properties. For example, existential quantification over the window size l bounded by $l_{\max}$ in $\mathsf{GW}_G(v, l_{\max})$ is preferable to only considering what happens in the window of size $l_{\max}$: the former option grants monotonicity, a natural wish for specification purposes, while the latter does not. On the same note, *undirect* variants of our objectives are prefix-independent, another interesting property.

Inductive property of windows. Let $\pi = s_0 s_1 s_2 \dots$ be a play. Fix any dimension t, $1 \leq t \leq k$. The window from position j to j', $0 \leq j < j'$, is *closed* if there exists j'', $j < j'' \leq j'$ such that the sum of weights in dimension t over the sequence $s_j \dots s_{j''}$ is non-negative. Otherwise the window is *open*. Given a position j' in π, a window is still open in j' if there exists a position $0 \leq j < j'$ such that the window from j to j' is open.

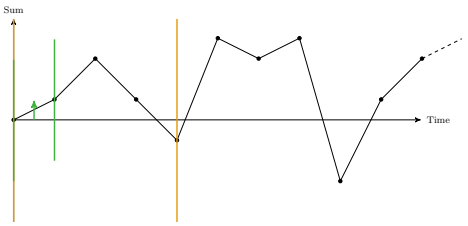
Consider any edge (s_i, s_{i+1}) appearing along π. If the edge is non-negative in dimension t, the window starting in i immediately closes. If not, a window opens that must be closed within $l_{\max}$ steps. Consider the *first* position i' such that this window closes, then we have that all intermediary opened windows also get closed by i', that is, for any i'', $i < i'' \leq i'$, the window starting in i'' is closed before or when reaching position i'. Indeed, the sum of weights over the window from i'' to i' is strictly greater than the sum over the window from i to i', which is non-negative. We call this fact the *inductive property of windows*.

8.2.2 Illustration

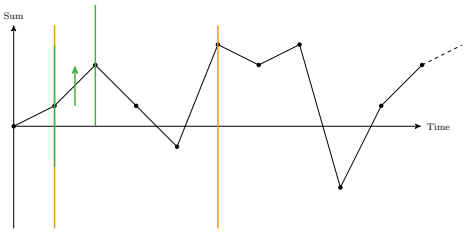
We depict the operation of the direct fixed window mean-payoff on a hypotetical play in Fig. 8.1. Furthermore, we provide two concrete examples in the following.



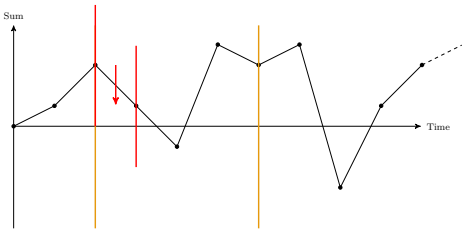
(a) The maximal window is placed over the initial state.



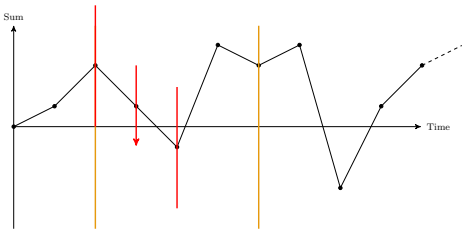
(b) The window of size 1 is good so the maximal window slides to the next state.



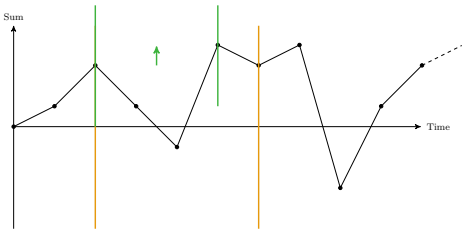
(c) Again, the first window is good and the maximal window slides.



(d) Window size 1 does not suffice: we enlarge the tested window to size 2.



(e) Still a bad window for size 2. But size 3 is still less than $l_{\max} = 4$, so we enlarge again.



(f) Finally, a non-negative mean-payoff is observed. The maximal window can resume sliding.

Figure 8.1: Illustration of the direct fixed window objective for maximal window size $l_{\max} = 4$ and threshold zero: the maximal window (orange) slides along the play from its starting state and a good window (green) should be found at each step. A window is good if the mean-payoff inside it is non-negative. Bad windows are in red. The tested window is enlarged incrementally up to the maximal window if necessary.

Example 8.1. Consider the game depicted in Fig. 8.2. It has a unique outcome, and it is winning for the classical mean-payoff objective of threshold 0, as well as for the infimum (resp. supremum) total-payoff objective of threshold -1 (resp. 0). Consider the fixed window mean-payoff objective for threshold 0. If the size of the window is bounded by 1, the play is losing.[3] However, if the window size is at least 2, the play is winning, as in s_3 we close the window in two steps and in s_4 in one step. Notice that by definition of the objective, it is clear that it is also satisfied for all larger sizes.[4] As the fixed window objective is satisfied for size 2, the bounded window objective is also satisfied. On the other hand, if we restrict the objectives to their direct variants, then none is satisfied, as from s_2, no window, no matter how large it is, gets closed. ◁

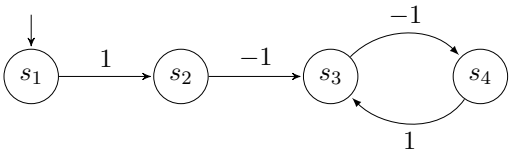


Figure 8.2: Fixed window is satisfied for $l_{\max} \geq 2$, whereas even direct bounded window is not.

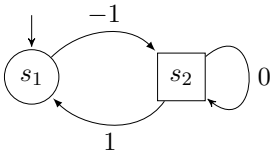


Figure 8.3: Mean-payoff is satisfied but none of the window objectives is.

Example 8.2. Consider the game of Fig. 8.3. Again, the unique strategy of $\mathcal{P}_1$ satisfies the mean-payoff objective for threshold 0 as all simple cycles are non-negative. It also ensures value -1 for the infimum and supremum total-payoffs. Consider the strategy of $\mathcal{P}_2$ that takes the self-loop once on the first visit of s_2, twice on the second, and so on. Clearly, it ensures that windows starting in s_1 stay open for longer and longer numbers of steps (we say that $\mathcal{P}_2$ *delays* the closing of the window), hence making the outcome losing for the bounded window objective (and thus the fixed window objective for any $l_{\max} \in \mathbb{N}_0$). This illustrates the added guarantee (compared to mean-payoff) asked by the window objective: in this case, no upper bound can be given on the time needed

[3] A window size of one actually requires that all infinitely often visited edges are of non-negative weights.

[4] As discussed before, the existential quantification on the window size l, bounded by $l_{\max}$, is indeed crucial in eq. (8.1) to ensure monotonicity with increasing maximal window sizes, a desired behavior of the definition for theoretical properties and intuitive use in specifications.

for a window to close, i.e., on the time needed to get the local sum back to non-negative. Note that $\mathcal{P}_2$ has to go back to s_1 at some point: otherwise, the prefix-independence of the objectives allows $\mathcal{P}_1$ to wait for $\mathcal{P}_2$ to settle on cycling and win. Also observe that $\mathcal{P}_2$ requires infinite-memory to continue to increase the delay unboundedly. For the direct variants, $\mathcal{P}_2$ has a simpler winning strategy consisting in looping forever, as enforcing one permanently open window is sufficient. ◁

8.3 Relation with Classical Objectives

We introduce the bounded window objectives as conservative approximations of mean-payoff and total-payoff in one-dimension games. Indeed, in Lemma 8.3, we show that winning the bounded window (resp. direct bounded window) objective implies winning the mean-payoff (resp. total-payoff) objective while the converse implication is only true if a strictly positive mean-payoff (resp. arbitrary high total-payoff) can be ensured.

Lemma 8.3. *Given a one-dimension game $G = (S_1, S_2, E, w)$, the following assertions hold.*

(a) If the answer to the bounded window mean-payoff problem is Yes*, then the answer to the mean-payoff threshold problem for threshold zero is also* Yes*.*

(b) If there exists $\varepsilon > 0$ such that the answer to the mean-payoff threshold problem for threshold ε is Yes*, then the answer to the bounded window mean-payoff problem is also* Yes*.*

(c) If the answer to the direct bounded window mean-payoff problem is Yes*, then the answer to the supremum total-payoff threshold problem for threshold zero is also* Yes*.*

(d) If the answer to the supremum total-payoff threshold problem is Yes *for all integer thresholds (i.e., the total-payoff value is ∞), then the answer to the direct bounded window mean-payoff problem is also* Yes*.*

Assertions *(a)* and *(c)* follow from the decomposition of winning plays into bounded windows of non-negative weights. The key idea for assertions *(b)*

and *(d)* is that mean-payoff and total-payoff objectives always admit *memoryless* winning strategies, for which the consistent outcomes can be decomposed into *simple cycles* (i.e., with no repeated edge) over which the mean-payoff is at least equal to the threshold and which length is bounded. Hence they correspond to closing windows. Note that strict equivalence with the classical objectives is not verified, as witnessed before (Fig. 8.3).

Proof. Assertion (a). In the one-dimension case, sup. and inf. mean-payoff problems coincide. Let $\pi \in \mathsf{Plays}(G)$ be such that $\pi \in \mathsf{BndWMP}_G(0)$. There exists $i \geq 0$ such that the suffix of π starting in i can be decomposed into an infinite sequence of bounded segments (i.e., windows) of non-negative weight. Thus, this suffix satisfies the sup. mean-payoff objective as there are infinitely many positions where the total sum from i is non-negative. Since the mean-payoff objective is prefix-independent, the play π is itself winning.

Assertion (b). Consider a memoryless winning strategy of $\mathcal{P}_1$ for the mean-payoff of threshold $\varepsilon > 0$. Only strictly positive simple cycles can be induced by such a strategy. Consider any outcome $\pi = s_0 s_1 s_2 \dots$ consistent with it. We claim that for any position j along this play, there exists a position $j + l$, with $l \leq l_{\max} = (|S| - 1) \cdot (1 + |S| \cdot W)$, such that the sum of weights over the sequence $\rho = s_j \dots s_{j+l}$ is non-negative. Clearly, if it is the case, then objective $\mathsf{FixWMP}_G(v, l_{\max})$ is satisfied and so is objective $\mathsf{BndWMP}_G(v)$. Consider the cycle decomposition $\mathcal{A}\mathcal{C}_1\mathcal{C}_2 \dots \mathcal{C}_n\mathcal{B}$ of this sequence obtained as follows. We push successively $s_0, s_1, \dots$ onto a stack, and whenever we push a state that is already in the stack, a simple cycle is formed that we remove from the stack and append to the cycle decomposition. The sequence ρ is decomposed into an acyclic part $(\mathcal{A} \cup \mathcal{B})$ of length[5] at most $(|S| - 1)$ and total sum at least $-(|S| - 1) \cdot W$ and simple cycles of total sum at least 1 and length at most $|S|$. Given the window size $l_{\max}$, we have at least $(|S| - 1) \cdot W$ simple cycles in the cycle decomposition. Hence, the total sum over ρ is at least zero, which proves our point.

Assertion (c). Consider a play $\pi \in \mathsf{DirBndWTP}_G(0)$. Using the same decomposition argument as for assertion *(a)*, we have that the sequence of total sums takes infinitely often values at least equal to zero. Thus the limit of this sequence of moments bounds from below the limit of the sequence of suprema and is at

[5]The length of a sequence is the number of *edges* it involves.

least equal to zero, which shows that the supremum total-payoff objective is also satisfied by play π.

Assertion (d). In one-dimension games, the value of the total-payoff (i.e., the largest threshold for which $\mathcal{P}_1$ has a winning strategy) is ∞ if and only if the value of mean-payoff is strictly positive [GS09]. Hence, we apply the argument of assertion *(b)*, further noticing that the window open in position j is closed in at most $l_{\max}$ steps for any $j \geq 0$, which is to say that the *direct* objective is satisfied. □

CHAPTER 9

One-Dimension Window Games

Fixed Window ◇ Bounded Window ◇ On Direct Objectives

For the *fixed window mean-payoff problem*, we establish an algorithm that runs in time polynomial in the size of the game and in the size of the window and we show that memory is needed for both players. Note that this is in contrast to the mean-payoff threshold problem, where $\mathcal{P}_2$ is memoryless even in the multi-dimension case (cf. Table 8.1 and Table 8.2). Moreover, the fixed window problem is shown to be P-hard even for polynomial window sizes.

For the *bounded window mean-payoff problem*, we prove equivalence with the fixed window problem for size $(|S| - 1) \cdot (|S| \cdot W + 1)$, i.e., this window size is sufficient to win if possible. The bounded window problem is then shown to be in NP $\cap$ coNP and at least as hard as mean-payoff games.

We conclude by adapting these results to the direct variants of the objectives.

This chapter is based on results published together with Chatterjee, Doyen and Raskin [CDRR13a, CDRR13b].

9.1 Fixed Window

9.1.1 Algorithm

Given a one-dimension game $G = (S_1, S_2, E, w)$ and a window size $l_{\max} \in \mathbb{N}_0$, we present an iterative algorithm FWMP (Alg. 9.1) to compute the winning states of $\mathcal{P}_1$ for the objective $\mathsf{FixWMP}_G(0, l_{\max})$.

Algorithm 9.1 FWMP$(G, l_{\max})$

Require: $G = (S_1, S_2, E, w)$ and $l_{\max} \in \mathbb{N}_0$
Ensure: W is the set of winning states for $\mathcal{P}_1$ for $\mathsf{FixWMP}_G(0, l_{\max})$

$n := 0$; $W := \emptyset$
repeat
  $W_d^n :=$ DirectFWMP$(G, l_{\max})$
  $W_{attr}^n := \mathsf{Attr}_G^{\mathcal{P}_1}(W_d^n)$ {attractor for $\mathcal{P}_1$}
  $W := W \cup W_{attr}^n$; $G := G \restriction (S \setminus W)$; $n := n + 1$
until $W = S$ or $W_{attr}^{n-1} = \emptyset$
return W

Initially, all states are potentially losing for $\mathcal{P}_1$. The algorithm iteratively declares states to be winning, removes them, and continues the computation on the remaining subgame as follows.

In every iteration, *i)* algorithm DirectFWMP computes the set of states W_d from which $\mathcal{P}_1$ can win the direct fixed window objective; *ii)* it computes the attractor to W_d; and then proceeds to the next iteration on the remaining subgame. States of the computed set W_d are obviously winning for the fixed window objective. Thanks to the prefix-independence of the fixed window objective, the attractor to W_d is also winning. Since $\mathcal{P}_2$ must avoid entering this attractor, $\mathcal{P}_2$ must restrict his choices to stay in the subgame, and hence we iterate on the remaining subgame. Thus states removed over all iterations are winning for $\mathcal{P}_1$.

The key argument to establish correctness is as follows: when the algorithm stops, the remaining set of states $\overline{W}$ is such that $\mathcal{P}_2$ can ensure to stay in $\overline{W}$ and falsify the direct fixed window objective by forcing the appearance of one open window larger than $l_{\max}$. Since he stays in $\overline{W}$, he can repeatedly use this strategy to falsify the fixed window objective. Thus the remaining set $\overline{W}$ is winning for $\mathcal{P}_2$, and the correctness of the algorithm follows.

Algorithm 9.2 DirectFWMP$(G, l_{\max})$

Require: $G = (S_1, S_2, E, w)$ and $l_{\max} \in \mathbb{N}_0$
Ensure: W_d is the set of winning states for $\mathcal{P}_1$ for $\mathsf{DirFixWMP}_G(0, l_{\max})$

$W_{gw} := \textsc{GoodWin}(G, l_{\max})$
if $W_{gw} = S$ or $W_{gw} = \emptyset$ **then**
  $W_d := W_{gw}$
else
  $W_d := \textsc{DirectFWMP}(G \restriction W_{gw}, l_{\max})$
return W_d

The main idea of algorithm DirectFWMP (Alg. 9.2) is that to win the direct fixed window objective, $\mathcal{P}_1$ must be able to repeatedly win the good window objective, which consists in ensuring a non-negative sum in at most $l_{\max}$ steps.

A winning strategy of $\mathcal{P}_1$ in a state s is thus a strategy that enforces a non-negative sum and, *as soon as the sum turns non-negative* (in some state s'), starts doing the same from s'. It is important to start again immediately as it ensures that all suffixes along the path from s to s' also have a non-negative sum thanks to the inductive property of windows. That is, for any state s'' in between, the window from s'' to s' is closed.

Algorithm 9.3 GoodWin$(G, l_{\max})$

Require: $G = (S_1, S_2, E, w)$ and $l_{\max} \in \mathbb{N}_0$
Ensure: W_{gw} is the set of winning states for $\mathsf{GW}_G(0, l_{\max})$

for all $s \in S$ **do**
  $C_0(s) := 0$
for all $i \in \{1, \dots, l_{\max}\}$ **do**
  for all $s \in S_1$ **do**
    $C_i(s) := \max_{(s,s') \in E}\{w((s, s')) + C_{i-1}(s')\}$
  for all $s \in S_2$ **do**
    $C_i(s) := \min_{(s,s') \in E}\{w((s, s')) + C_{i-1}(s')\}$
return $W_{gw} := \{s \in S \mid \exists\, i,\ 1 \leq i \leq l_{\max},\ C_i(s) \geq 0\}$

The set of states from which $\mathcal{P}_1$ can ensure winning for the good window objective is computed by subroutine GoodWin (Alg. 9.3). Intuitively, given a state $s \in S$ and a number of steps $i \geq 1$, the value $C_i(s)$ is computed iteratively (from $C_{i-1}(s)$) and represents the best sum that $\mathcal{P}_1$ can ensure from s in exactly i steps. Hence, the set of winning states for $\mathcal{P}_1$ is the set of states for which there

exists some i, $1 \leq i \leq l_{\max}$ such that $C_i(s) \geq 0$. We state the correctness of GoodWin in Lemma 9.1.

Lemma 9.1. *Algorithm* GoodWin *computes the set of winning states of* $\mathcal{P}_1$ *for the good window objective in time* $\mathcal{O}(|S| \cdot |E| \cdot l_{\max} \cdot V)$*, with* $V = \lceil \log_2 W \rceil$*, the length of the binary encoding of weights.*

Proof. Let $\mathcal{W}_g \subseteq S$ denote the winning states for $\mathsf{GW}_G(0, l_{\max})$. We prove that (a) $s \in \mathcal{W}_g \Rightarrow s \in \textsc{GoodWin}(G, l_{\max})$, and (b) $s \in \textsc{GoodWin}(G, l_{\max}) \Rightarrow s \in \mathcal{W}_g$.

We first consider case (a). From s, there exists a strategy of $\mathcal{P}_1$ that enforces a non-negative sum after l steps, for some l, $1 \leq l \leq l_{\max}$. Hence, the value $C_l(s)$ computed by the algorithm is non-negative and $s \in \textsc{GoodWin}(G, l_{\max})$.

Case (b). Assume $s \in \textsc{GoodWin}(G, l_{\max})$. By definition of the algorithm GoodWin, there exists some $l \leq l_{\max}$ such that $C_l(s)$ is positive. Consequently, taking the choice of l edges that achieves the maximum value defines a strategy for $\mathcal{P}_1$ that ensures a positive sum after l steps, hence closing the window started in s. That is, $s \in \mathcal{W}_g$.

It remains to discuss the complexity of GoodWin. Clearly, it takes a number of elementary arithmetic operations which is bounded by $\mathcal{O}(|S| \cdot |E| \cdot l_{\max})$ to compute the set W_{gw}. Each elementary arithmetic operation takes time linear in the number of bits V of the encoding of weights, that is, logarithmic in the largest weight W. Hence, the time complexity of GoodWin is $\mathcal{O}(|S| \cdot |E| \cdot l_{\max} \cdot V)$. □

Thanks to the previous lemma, we establish the algorithm solving the direct fixed window objective.

Lemma 9.2. *Algorithm* DirectFWMP *computes the winning states of* $\mathcal{P}_1$ *for the direct fixed window mean-payoff objective in time* $\mathcal{O}\left(|S|^2 \cdot |E| \cdot l_{\max} \cdot V\right)$*, with* $V = \lceil \log_2 W \rceil$*, the length of the binary encoding of weights.*

Proof. Let $\mathcal{W}$ be the set of winning states for $\mathsf{DirFixWMP}_G(0, l_{\max})$, i.e.,

$$s \in \mathcal{W} \;\Leftrightarrow\; \exists\, \lambda_1 \in \Lambda_1,\ \forall\, \lambda_2 \in \Lambda_2,\ \mathsf{Outs}_G(s, \lambda_1, \lambda_2) \in \mathsf{DirFixWMP}_G(0, l_{\max}).$$

We first prove (a) $s \in \textsc{DirectFWMP}(G, l_{\max}) \Rightarrow s \in \mathcal{W}$, and then (b) $s \in \mathcal{W} \Rightarrow s \in \textsc{DirectFWMP}(G, l_{\max})$. First of all, notice that DirectFWMP exactly

computes the set of states W_d such that a non-negative sum is achievable in at most $l_{\max}$ steps, using only states from which a non-negative sum can also be achieved in at most $l_{\max}$ steps (hence the property is defined recursively).

Consider case (a). Let $s \in W_d$. Consider the following strategy of $\mathcal{P}_1$.

1. Play the strategy prescribed by GoodWin until a non-negative sum is reached. This is guaranteed to be the case in at most $l_{\max}$ steps. Let s' be the state that is reached in this manner.
2. By construction of W_d, we have that $s' \in W_d$. Thus, play the strategy prescribed by GoodWin in s'.
3. Continue ad infinitum.

We denote this strategy by λ_1 and claim it is winning for the direct fixed window objective, i.e., $s \in \mathcal{W}$. Indeed, consider any strategy of $\mathcal{P}_2$ and let $\pi = \mathsf{Outs}_G(s, \lambda_1, \lambda_2)$. We have $\pi = \sigma_1\sigma_2 \dots \sigma_{m_1}\sigma_{m_1+1} \dots \sigma_{m_2}\sigma_{m_2+1} \dots$ with $\forall\, j \geq 0$, $\sigma_j \in S$ and $\sigma_1 = \sigma_{m_0} = s$, such that all sequences $\rho(n) = \sigma_{m_n} \dots \sigma_{m_{n+1}}$ are of length at most $l_{\max} + 1$ ($l_{\max}$ steps) and such that all strict prefixes of $\rho(n)$ are strictly negative and all suffixes of $\rho(n)$ are positive. Indeed, starting in some state σ_{m_n}, the strategy λ_1 keeps a memory of the current sum and tries to reach a non-negative value (using the strategy prescribed by GoodWin). As soon as such a value is reached in a state $\sigma_{m_{n+1}}$, the memory of the current sum kept by the strategy is reset to zero and the process is restarted. That way, for all j, $m_n \leq j < m_{n+1}$, we have that the sum over the sequence from σ_j to $\sigma_{m_{n+1}}$ is non-negative, hence all intermediate windows are also closed. Thus, the window property is satisfied everywhere along the play π, starting in $\sigma_1 = s$, which proves that $s \in \mathcal{W}$.

Case (b). Let λ_1 be a winning strategy of $\mathcal{P}_1$ for $\mathsf{DirFixWMP}_G(0, l_{\max})$. For any strategy λ_2 of $\mathcal{P}_2$, the outcome is a play $\pi = \sigma_1\sigma_2 \dots$ with $\sigma_1 = s$ such that the window property is satisfied from all states. In particular, this implies, that for all σ_j, strategy λ_1 enforces a positive sum in at most $l_{\max}$ steps, that is, $\sigma_j \in \textsc{GoodWin}(G, l_{\max})$. Since it is the case for all states σ_j, we have that $\mathcal{P}_1$ has a strategy to ensure a positive sum in at most $l_{\max}$ steps using only states from which this property is ensured. Therefore, we conclude that $s \in W_d$.

Again, the number of calls is at most the number of states $|S|$. Let $\mathbb{C}_{\mathsf{GW}}$ denote the complexity of algorithm GoodWin. Then, the time complexity of

algorithm DirectFWMP is $\mathcal{O}\left(|S| \cdot \mathbb{C}_{\mathsf{GW}}\right)$. □

Finally, we prove correctness of the algorithm for the fixed window problem.

Lemma 9.3. *Algorithm* FWMP *computes the set of winning states of* $\mathcal{P}_1$ *for the fixed window mean-payoff objective in time* $\mathcal{O}\left(|S|^3 \cdot |E| \cdot l_{\max} \cdot V\right)$*, with* $V = \lceil \log_2 W \rceil$*, the length of the binary encoding of weights.*

Proof. Let $\mathcal{W} \subseteq S$ be the set of states that are winning for $\mathsf{FixWMP}_G(0, l_{\max})$, i.e.,

$$s \in \mathcal{W} \;\Leftrightarrow\; \exists\, \lambda_1 \in \Lambda_1,\ \forall\, \lambda_2 \in \Lambda_2,\ \mathsf{Outs}_G(s, \lambda_1, \lambda_2) \in \mathsf{FixWMP}_G(0, l_{\max}).$$

Note that since we set the threshold to be 0 (w.l.o.g.), we may ignore the division by the window size l in eq. (8.1). We claim that $\mathrm{FWMP}(G, l_{\max}) = \mathcal{W}$. The proof is in two parts: (a) $s \in \mathrm{FWMP}(G, l_{\max}) \Rightarrow s \in \mathcal{W}$, and (b) $s \in \mathcal{W} \Rightarrow s \in \mathrm{FWMP}(G, l_{\max})$.

We begin with (a). Let $(W_d)^{n \geq 0}$ and $(W_{attr})^{n \geq 0}$ be the finite sequences of sets computed by the iterative algorithm. We have that $\mathrm{FWMP}(G, l_{\max}) = \bigcup_{n \geq 0} W^n_{attr}$. For any n, n' such that $n \neq n'$, we have that $W^n_{attr} \cap W^{n'}_{attr} = \emptyset$ and $W^n_d \cap W^{n'}_d = \emptyset$. Moreover, for all $n \geq 0$, $W^n_d \subseteq W^n_{attr}$. Let $s \in \mathrm{FWMP}(G, l_{\max})$. There exists a unique $n \geq 0$ such that $s \in W^n_{attr}$. By construction, from s, $\mathcal{P}_1$ has a strategy to reach and stay in $W^n_d \cup W^{n-1}_{attr} \cup W^{n-2}_{attr} \cup \ldots W^0_{attr}$ and thus s is winning in the subgame $G \restriction (S \setminus W^{n-1}_{attr})$. However, $\mathcal{P}_2$ still has the possibility to leave W^n_d and reach the set $W^{n-1}_{attr} \cup W^{n-2}_{attr} \cup \ldots W^0_{attr}$. Since the sequence is finite and $\mathcal{P}_2$ cannot leave W^0_d, we have that at some point, any outcome is trapped in some set W^m_d, $0 \leq m \leq n$, in which $\mathcal{P}_1$ wins the direct fixed window objective. Let x be the length of the finite prefix outside W^m_d. The outcome satisfies the fixed window mean-payoff objective for $i = x$. Therefore, we have that $s \in \mathcal{W}$.

Now consider (b). Let $s \in \mathcal{W}$ be a winning state for $\mathsf{FixWMP}_G(0, l_{\max})$. We claim that $s \in \mathrm{FWMP}(G, l_{\max})$. Suppose it is not the case and consider the sequences $(W_d)^{n \geq 0}$ and $(W_{attr})^{n \geq 0}$ as before. We have that for all $n \geq 0$, $s \notin W^n_{attr}$. In particular, $\mathcal{P}_2$ can force staying in $S_{trap} = S \setminus \bigcup_{n \geq 0} W^n_{attr}$ when starting in s. Since the algorithm has stopped, we have $\mathrm{DirectFWMP}(G \restriction S_{trap}, l_{\max}) = \emptyset$. As algorithm DirectFWMP is correct, from all states of S_{trap}, $\mathcal{P}_2$ has a strategy to spoil the direct fixed window game, i.e., $\mathcal{P}_2$ can force a sequence of states

such that there exists a position j along it for which the window starting in j stays open for at least $(l_{\max} + 1)$ steps, and such that this sequence remains in S_{trap}. Therefore, $\mathcal{P}_2$ can force staying in S_{trap} and seeing infinitely often such sequences, hence $\mathcal{P}_1$ is losing for the fixed window mean-payoff objective, which contradicts the fact that $s \in \mathcal{W}$.

Finally, consider the complexity of the recursive algorithm FWMP. Notice that at least one state is declared winning at each iteration. The number of calls is thus at most the number of states $|S|$. Computing the attractor is linear in the number of edges $|E| \leq |S|^2$. The overall complexity is thus $\mathcal{O}\left(|S| \cdot (|E| + \mathbb{C}_{\mathsf{DW}})\right)$, where $\mathbb{C}_{\mathsf{DW}}$ is the complexity of the DirectFWMP algorithm. □

9.1.2 Memory and Complexity Bounds

Thanks to the correctness of algorithm FWMP, we also deduce linear upper bounds (in $|S| \cdot l_{\max}$) on the memory needed for both players (Lemma 9.4). Indeed, let $s \in S$ be a winning state for $\mathcal{P}_1$. A winning strategy λ_1 for $\mathcal{P}_1$ is to (a) reach the set of states W_d^n that are winning for the direct fixed window objective in the subgame restricted to states $W_d^n \setminus W_{attr}^{n-1}$, then (b) repeatedly play the strategy prescribed by GoodWin in this subgame (i.e., enforce a non-negative sum in less than $l_{\max}$ steps, see proof of Lemma 9.2). If $\mathcal{P}_2$ leaves for a lower subgame restricted to $W_{attr}^{n'}$, $n' < n$, the strategy is to start again part (a) in this subgame. Part (a) is memoryless as it uses a classical attractor strategy. Part (b) requires to consider, for each state s' in the set computed by DirectFWMP, a number of memory states which is bounded by $l_{\max}$, as the only memory needed is to select the corresponding successor state that will maximize the $C_l(s')$ value, for all possible values of l, the number of steps remaining to close a window. Similarly, $\mathcal{P}_2$ needs to be able to prevent the closing of a window repeatedly, and therefore also possibly needs $l_{\max}$ memory states for each state of the game.

To illustrate that memory is needed by both players, consider the following examples. First, consider a game where all states belong to $\mathcal{P}_1$ and such that the play starts in a central state s and in s, there are three outgoing edges, toward three simple cycles $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$. All other states have only one outgoing edge. Cycle $\mathcal{C}_1$ is composed of six edges of successive weights $3, 3, 5, -1, -1$ and -5. Cycle $\mathcal{C}_2$ is $7, -1$ and -9. Cycle $\mathcal{C}_3$ is $5, 5$ and -11. The objective is $\mathsf{FixWMP}_G(0, l_{\max} = 4)$. Clearly, from some point on, a winning strategy of $\mathcal{P}_1$ has

to infinitely alternate between cycles in the following way: $(\mathcal{C}_1\mathcal{C}_2\mathcal{C}_3)^\omega$. Any other alternation leads to a bad window appearing infinitely often: hence, the decision of $\mathcal{P}_1$ in s depends on the remaining number of steps to ensure a good window. Second, consider a similar game but with all states belonging to $\mathcal{P}_2$. Again, the initial state is central and there are two cycles $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\mathcal{C}_1$ is 1 followed by -1, and $\mathcal{C}_2$ is $-1, -1$ and 2. The objective is $\mathsf{FixWMP}_G(0, l_{\max} = 3)$. If $\mathcal{P}_2$ is memoryless, both possible strategies induce a winning play for $\mathcal{P}_1$. On the other hand, if $\mathcal{P}_2$ is allowed to alternate, he can choose the play $(\mathcal{C}_1\mathcal{C}_2)^\omega$ which will be losing for $\mathcal{P}_1$ as the window $-1, -1, -1$ will appear infinitely often.

Lemma 9.4. *In one-dimension games with a fixed window mean-payoff objective, memory is needed by both players and linear memory in the number of states times the window size is sufficient.*

Through Lemma 9.3, we have shown that the fixed window problem admits a polynomial (in $|S|$, V and $l_{\max}$) algorithm. In Lemma 9.5, we prove that even for window size $l_{\max} = 1$ and weights $\{-1, 1\}$, the problem is P-hard. This is via a reduction from reachability games. By making the target states absorbing with a self-loop of weight 1, and giving weight -1 on all other edges, we obtain the reduction, as reaching a target state is now the only way to ensure that windows close.

Lemma 9.5. *In two-player one-dimension games, the fixed window mean-payoff problem is P-hard, even for $l_{\max} = 1$ and weights $\{-1, 1\}$.*

Proof. Let $G_r = (S_1, S_2, E)$ be an unweighted game with a reachability objective asking to visit (at least once) a state of the set $R \subseteq S$. We build the game $G = (S_1, S_2, E', w)$ by (a) making the target states absorbing with a self-loop of weight 1, i.e., for all $s \in R$, we have $(s, s) \in E'$ and $w((s, s)) = 1$, and (b) putting weight -1 on all other edges, i.e., for all edge $(s, t) \in E$ such that $s \notin R$, we have $(s, t) \in E'$ and $w((s, s)) = -1$. We claim that $\mathcal{P}_1$ has a winning strategy in G_r from a state $s \in S$ if and only if he has a winning strategy for the objective $\mathsf{FixWMP}_G(0, l_{\max} = 1)$ in G from $s \in S$. Indeed, it is clear that any outcome that never reaches the target set is such that all windows stay indefinitely open, and conversely, an outcome that reaches this set after n steps is winning for the fixed window objective with $i = n$. Since deciding the winner in reachability games is P-complete [Bee80, Imm81], this concludes our proof. □

9.1.3 Wrap-up

We sum up the complexity analysis of the fixed window problem in Theorem 9.6.

Theorem 9.6. *In two-player one-dimension games, (a) the fixed arbitrary window mean-payoff problem is decidable in time* $\mathcal{O}\left(|S|^3 \cdot |E| \cdot l_{\max} \cdot V\right)$*, with* $V = \lceil \log_2 W \rceil$*, the length of the binary encoding of weights, and (b) the fixed polynomial window mean-payoff problem is P-complete. In general, both players require memory, and memory of size linear in* $|S| \cdot l_{\max}$ *is sufficient.*

9.2 Bounded Window

9.2.1 Algorithm

In the following, we focus on the bounded window mean-payoff problem for two-player one-dimension games. We start with two technical lemmas related to the classical supremum total-payoff threshold problem. Using these lemmas, we establish an NP ∩ coNP algorithm to solve the bounded window problem and, as a corollary, we get an interesting bound on the window size needed to win the fixed window problem if possible.

The first technical lemma (Lemma 9.7) states that if $\mathcal{P}_1$ has a strategy to win the supremum total-payoff objective from some state s_{init}, then he can force a non-negative sum from this state in at most $(|S|-1) \cdot (|S| \cdot W + 1)$ steps, i.e., he wins the good window objective for this window size.

Lemma 9.7. *Let* $G = (S_1, S_2, E, w)$ *be a two-player one-dimension game. If* $\mathcal{P}_1$ *has a strategy to win for objective* $\mathsf{TotalSup}_G(0)$ *from initial state* $s_{\mathsf{init}} \in S$*, then* $\mathcal{P}_1$ *also has a strategy to win for the good window objective* $\mathsf{GW}_G(0, l_{\max})$ *from* s_{init} *for* $l_{\max} = (|S|-1) \cdot (|S| \cdot W + 1)$*.*

This result is obtained by considering a memoryless winning strategy of $\mathcal{P}_1$ for the total-payoff and the decomposition in simple cycles of any consistent outcome where (a) either simple cycles are strictly positive, or (b) they are of value zero but preceded by a non-negative prefix.

Proof. Let $\lambda_1 \in \Lambda_1^{PM}$ be a memoryless winning strategy of $\mathcal{P}_1$ for $\mathsf{TotalSup}_G(0)$. Our claim is that for all possible outcome π consistent with λ_1 starting in the

initial state s_{init}, there exists a prefix ρ of π of size at most $l_{\max}$ such that the total sum of weights over ρ is non-negative. Let π be any outcome consistent with λ_1 and ρ_1 its prefix of length $(|S|-1)\cdot(|S|\cdot W+1)$. Consider the cycle decomposition (see the proof of Lemma 8.3) of ρ_1: $\mathcal{A}, \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m, \mathcal{B}$, with $\mathcal{A}$ the prefix before the first cycle and $\mathcal{B}$ the suffix after the last cycle in ρ_1. The total length of the acyclic part is $|\mathcal{A}|+|\mathcal{B}| < |S|-1$. We claim that there exists a prefix ρ of ρ_1 such that the total sum of weights over ρ is non-negative. Consider the following arguments:

1. No cycle $\mathcal{C}$ in $\{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ can be strictly negative. Otherwise, since λ_1 is memoryless, $\mathcal{P}_2$ could force cycling in such a cycle forever and the play would be losing for the supremum total-payoff objective, which contradicts λ_1 being a winning strategy.

2. Assume that there exists a cycle $\mathcal{C}$ in $\{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ such that the sum of weights over this cycle is zero. We define the *high point* of a cycle as the first state where the sum from the start of the cycle takes its highest value. Then, the prefix ρ of ρ_1 up to this high point is non-negative and we are done. Indeed, assume it is not the case. Then, the running sum over the outcome π is strictly negative when reaching the high point, and stays strictly negative in all positions along the cycle $\mathcal{C}$, by definition of the high point. Therefore, $\mathcal{P}_2$ can force cycling forever in $\mathcal{C}$ since λ_1 is memoryless and the outcome becomes losing for the total-payoff objective.

3. So assume there are only strictly positive cycles in the cycle decomposition of ρ_1, that is, they all have a total sum of value at least 1. The total sum over $\mathcal{C}_1, \ldots, \mathcal{C}_m$ is at least equal to m. Since each cycle is of length at most $|S|$ and $\mathcal{A} \cup \mathcal{B}$ is of length at most $|S|-1$, we have that the number of cycles m in the cycle decomposition of ρ_1 is at least

$$((|S|-1)\cdot(|S|\cdot W+1) - (|S|-1)) \,/\, |S| = (|S|-1)\cdot W.$$

   Given that the total sum over prefix $\mathcal{A}$ is at least $-(|S|-1)\cdot W$, we obtain that $\rho = \mathcal{A}\,\mathcal{C}_1\,\mathcal{C}_2 \ldots \mathcal{C}_m$ is the desired prefix with a non-negative total sum, and its length is bounded by $(|S|-1)\cdot(|S|\cdot W+1)$.

This concludes our proof. □

The second technical lemma (Lemma 9.8) shows that if $\mathcal{P}_2$ has a strategy to ensure that the supremum total-payoff from some state s_{init} is strictly negative, then he has a memoryless strategy to do so and any outcome π starting in s_{init} and consistent with this strategy is such that the direct bounded window mean-payoff objective is not satisfied.

Lemma 9.8. *Let $G = (S_1, S_2, E, w)$ be a two-player one-dimension game. If $\mathcal{P}_2$ has a spoiling strategy for objective $\mathsf{TotalSup}_G(0)$ from initial state $s_{\mathsf{init}} \in S$, then $\mathcal{P}_2$ has a strategy $\lambda_2 \in \Lambda_2^{PM}$ to ensure that for all outcome $\pi = \sigma_0\sigma_1\dots$ consistent with λ_2 starting in $\sigma_0 = s_{\mathsf{init}}$, there exists a position $i \geq 0$ such that for all window sizes $l \geq 1$, the total sum of weights on the window from σ_i to σ_{i+l} is strictly negative.*

Proof. By contradiction. Let $\lambda_2 \in \Lambda_2^{PM}$ be a memoryless spoiling strategy for objective $\mathsf{TotalSup}_G(0)$ from $s_{\mathsf{init}} \in S$. Let π be a consistent outcome and assume that it does not respect the lemma, i.e., for all positions $i \geq 0$, there exists a window size $l \geq 1$ such that the window from σ_i to σ_{i+l} is non-negative. Then the play π can be decomposed as a sequence of finite windows of non-negative weights. Hence, the total sum from $\sigma_0 = s_{\mathsf{init}}$ takes infinitely often values at least equal to zero and the limit of its suprema is non-negative. This is in contradiction to λ_2 being a winning strategy for $\mathcal{P}_2$. □

Thanks to Lemma 9.7 and Lemma 9.8, we are now able to establish an NP ∩ coNP algorithm (Alg. 9.4) to solve the bounded window mean-payoff problem on two-player one-dimension games. Lemma 9.9 states its correctness.

Algorithm BoundedProblem (Alg. 9.4) first computes via a subroutine named UnbOpenWindow the set of states from which $\mathcal{P}_2$ can force the visit of a position such that the window opening in this position never closes.

Clearly, to prevent $\mathcal{P}_1$ from winning the bounded window problem, $\mathcal{P}_2$ must be able to do so repeatedly as the prefix-independence of the objective otherwise gives the possibility to wait that all such bad positions are encountered before taking the windows into account. Therefore, the states that are not in UnbOpenWindow(G), as well as their attractor, are winning for $\mathcal{P}_1$. Since the choices of $\mathcal{P}_2$ are reduced because of the attractor of $\mathcal{P}_1$ being declared winning, we compute in several steps, adding new states to the set of winning states for $\mathcal{P}_1$ up to stabilization.

Algorithm 9.4 BOUNDEDPROBLEM(G)

Require: Game $G = (S_1, S_2, E, w)$
Ensure: W_{bp} is the set of winning states for $\mathcal{P}_1$ for the bounded window mean-payoff problem
$W_{bp} := \emptyset$
$L :=$ UNBOPENWINDOW(G)
while $L \neq S \setminus W_{bp}$ **do**
$W_{bp} := \mathsf{Attr}_G^{\mathcal{P}_1}(S \setminus L)$
$L :=$ UNBOPENWINDOW$\Big(G \restriction (S \setminus W_{bp})\Big)$
return W_{bp}

Algorithm 9.5 UNBOPENWINDOW(G)

Require: Game $G = (S_1, S_2, E, w)$
Ensure: L is the set of states from which $\mathcal{P}_2$ can force a position for which the window never closes
$p := 0$; $L_0 := \emptyset$
repeat
$L_{p+1} := L_p \cup \mathsf{Attr}^{\mathcal{P}_2}_{G\restriction(S\setminus L_p)}\Big(\text{NEGSUPTP}\big(G \restriction (S \setminus L_p)\big)\Big)$
$p := p + 1$
until $L_p = L_{p-1}$
return $L := L_p$

Now consider the subroutine UNBOPENWINDOW (Alg. 9.5). Its correctness is based on Lemma 9.8. Indeed, it computes the set of states from which $\mathcal{P}_2$ can force a position for which the window never closes. To do so, it suffices to compute the attractor for $\mathcal{P}_2$ of the set of states from which $\mathcal{P}_2$ can enforce a strictly negative supremum total-payoff. Routine NEGSUPTP returns this set of states in NP $\cap$ coNP complexity [GS09]. Again, we compute the fixed point of the sequence as at each iteration, the choices of $\mathcal{P}_1$ are reduced.

The main idea of the correctness proof is that from all states in $\overline{W_{bp}}$, $\mathcal{P}_2$ has an infinite-memory winning strategy which is played in rounds, and in round n ensures an open window of size at least n by playing the total-payoff strategy of $\mathcal{P}_2$ for at most $n \cdot |S|$ steps, and then proceeds to round $(n+1)$ to ensure an open window of size $(n+1)$, and so on. Hence, windows stay open for arbitrary large periods and the bounded window objective is falsified.

Lemma 9.9. *Given a two-player one-dimension game $G = (S_1, S_2, E, w)$, algorithm* BoundedProblem *computes the set of winning states for $\mathcal{P}_1$ for the bounded window mean-payoff objective of threshold 0 in time $\mathcal{O}(|S|^2 \cdot (|E| + \mathbb{C}))$, where $\mathbb{C}$ is the complexity of algorithm* NegSupTP*, i.e., the complexity of computing the set of winning states in a two-player one-dimension supremum total-payoff game. Thus, algorithm* BoundedProblem *is in NP ∩ coNP.*

Proof. We show that for all states in $W_{bp} = \text{BoundedProblem}(G)$, there exists a winning strategy of $\mathcal{P}_1$, whereas for all states in $S \setminus W_{bp}$, there exists one of $\mathcal{P}_2$.

Consider a state $s \in W_{bp}$. Consider $(L^m)_{0 \leq m \leq n}$, the finite sequence of sets L that are computed by BoundedProblem, with $L_0 = \text{UnbOpenWindow}(G)$; and $(W_{bp}^m)_{0 \leq m \leq n}$, the corresponding finite sequence of sets W_{bp} where $W_{bp}^0 = \emptyset$ is empty and $W_{bp}^n = W_{bp}$ is the returned set of winning states. For all m', m, $0 \leq m' < m \leq n$, we have that $W_{bp}^m \supset W_{bp}^{m'}$ and $L^m \subset L^{m'}$. By construction, there exists m, $1 \leq m \leq n$ such that $s \in W_{bp}^m = \mathsf{Attr}_G^{\mathcal{P}_1}(S \setminus L^{m-1})$. In the subgame $G \restriction ((S \setminus L^{m-1}) \setminus W_{bp}^{m-1})$, $\mathcal{P}_1$ has a memoryless [GZ04] winning strategy for the supremum total-payoff objective. Hence, consider the strategy λ_1 of $\mathcal{P}_1$ which is to reach the set $(S \setminus L^{m-1})$ (in at most $|S|$ steps) and then play the memoryless total-payoff strategy in the subgame. It is possible for $\mathcal{P}_2$ to force leaving this subgame for a lower subset $W_{bp}^{m'} \subset W_{bp}^m$ with $m' < m$ but since the sequence is finite, any outcome is ultimately trapped in some subgame $G \restriction ((S \setminus L^{m''}) \setminus W_{bp}^{m''})$. Therefore, repeating the strategy λ_1 in each subgame ensures that after a finite number of steps (and hence a finite number of positions for which windows never close), a bottom subgame $G \restriction ((S \setminus L^{m''}) \setminus W_{bp}^{m''})$ is reached and, by Lemma 9.7, strategy λ_1 ensures satisfaction of the good window objective for $l_{\max} = (|S| - 1) \cdot (|S| \cdot W + 1)$ in this subgame. Moreover, since this strategy never visits states out of the bottom subgame, it ensures an inductive window from every state, regardless of the past. Hence, all intermediate windows are also closed and this strategy is winning for $\mathsf{FixWMP}_G(0, l_{\max}) \subseteq \mathsf{BndWMP}_G(0)$ from the initial state s. The states that are only visited finitely often before reaching the bottom subgame have no consequence thanks to the prefix-independence of the bounded window mean-payoff objective.

As for $\mathcal{P}_2$, consider a state $s \in S \setminus W_{bp}$. Consider $(L_p)_{0 \leq p \leq q}$, the finite sequence of sets L that are computed in the last call to UnbOpenWindow by BoundedProblem, with $L_0 = \emptyset$. We define the sequences $(N_p)_{1 \leq p \leq q}$

and $(A_p)_{1\leq p\leq q}$ as $N_p = \text{NegSupTP}(G \restriction (S \setminus L_{p-1}))$ and $A_p = L_p \setminus L_{p-1} = \mathsf{Attr}^{\mathcal{P}_2}_{G\restriction(S\setminus L_{p-1})}(N_p)$. We have that $s \in L_p$ for some p between 1 and q. An infinite memory winning strategy for $\mathcal{P}_2$ is played in rounds. In round n, $\mathcal{P}_2$ acts as follows. (a) If the current state is in A_p, play the attractor to N_p and then play the optimal strategy for the supremum total-payoff in N_p to ensure that no window will have a non-negative sum for n steps. (b) $\mathcal{P}_1$ can leave the set A_p for some lower set $A_{p'}$, $1 \leq p' < p$. If so, play the attractor to $N_{p'}$ and continue. Ultimately, any outcome is trapped in some set $N_{p''} \setminus A_{p''-1}$, with $1 \leq p'' \leq q$ and $A_0 = \emptyset$, as in N_1, $\mathcal{P}_1$ cannot leave. There $\mathcal{P}_1$ cannot prevent the window being strictly negative for n steps. When such a window has been enforced for n steps, move to round $n+1$ and start again. This strategy ensures that the bounded window problem is not satisfied as, infinitely often, windows stay open for arbitrary large periods along any outcome.

Finally, we discuss the complexity of algorithm BoundedProblem. Let $\mathbb{C}$ be the complexity of routine NegSupTP, that is, the complexity of solving a one-dimension supremum total-payoff game. The total complexity of subalgorithm UnbOpenWindow is $\mathcal{O}(|S| \cdot (|E| + \mathbb{C}))$ as the sequence of computations is of length at most $|S|$ and each computation takes time $\mathcal{O}(|E| + \mathbb{C})$. The overall complexity of BoundedProblem is thus $\mathcal{O}(\mathbb{C} + |S| \cdot (|E| + |S| \cdot (|E| + \mathbb{C}))) = \mathcal{O}(|S|^2 \cdot (|E| + \mathbb{C}))$. □

An interesting corollary of Lemma 9.7 and Lemma 9.9 is that sets of winning states coincide for objectives $\mathsf{FixWMP}_G(0, l_{\max} = (|S|-1)\cdot(|S|\cdot W+1))$ and $\mathsf{BndWMP}_G(0)$, therefore proving NP ∩ coNP-membership for the subset of fixed window problems with size at least $l_{\max}$ (hence an algorithm independent of the window size whereas Lemma 9.2 gives an algorithm which is polynomial in it).

Corollary 9.10. *In two-player one-dimension games, the fixed window mean-payoff problem is in NP ∩ coNP for window size at least equal to* $(|S|-1)\cdot(|S|\cdot W+1)$.

9.2.2 Memory and Complexity Bounds

Algorithm BoundedProblem (Lemma 9.9) gives memoryless strategies for $\mathcal{P}_1$ (attractor + memoryless strategy for total-payoff) and infinite-memory strategies for $\mathcal{P}_2$ (delaying the closing of windows for increasing number of steps each round)

in one-dimension bounded window mean-payoff games. Lemma 9.11 states that infinite-memory strategies are necessary for $\mathcal{P}_2$, as discussed in Ex. 8.2: $\mathcal{P}_2$ cannot use the zero cycle forever, but he must cycle long enough to defeat any finite window. Hence, its strategy needs to cycle for longer and longer, which requires infinite memory.

Lemma 9.11. *In one-dimension games with a bounded window mean-payoff objective, (a) memoryless strategies suffice for $\mathcal{P}_1$, and (b) infinite-memory strategies are needed for $\mathcal{P}_2$ in general.*

In Lemma 9.14, we give a polynomial reduction from mean-payoff games to bounded window mean-payoff games, therefore showing that a polynomial algorithm for the bounded window problem would solve the long-standing question of the P-membership of the mean-payoff threshold problem. The proof relies on technical lemmas providing intermediary reductions. First, we prove that given a game G, deciding if $\mathcal{P}_1$ has a strategy to ensure a non-negative mean-payoff can be reduced to deciding if $\mathcal{P}_1$ has a strategy to ensure a strictly positive mean-payoff when weights are shifted positively by a sufficiently small ε (Lemma 9.12). Second, we apply Lemma 8.3 on the shifted game to prove that winning this objective implies winning the bounded window problem. This gives one direction of the reduction. For the other one, we show that given a game G, if $\mathcal{P}_1$ has a strategy to win the bounded window problem when weights are shifted positively by a sufficiently small ε, he has one to win the mean-payoff threshold problem in G.

We define the following notation: given a two-player one-dimension game $G = (S_1, S_2, E, w)$ and $\varepsilon \in \mathbb{Q}$, let $G_{+\varepsilon} = (S_1, S_2, E, w_{+\varepsilon})$ be the game obtained by shifting all weights by ε, that is, for all $e \in E$, $w_{+\varepsilon}(e) = w(e) + \varepsilon$.[1]

Lemma 9.12. *For all one-dimension game $G = (S_1, S_2, E, w)$ with integer weights, for all ε, $0 < \varepsilon < 1/|S|$, for all initial state $s \in S$, $\mathcal{P}_1$ has a strategy to ensure a non-negative mean-payoff in G if and only if $\mathcal{P}_1$ has a strategy to ensure a strictly positive mean-payoff in $G_{+\varepsilon}$.*

Proof. Consider a memoryless winning strategy of $\mathcal{P}_1$ in G from initial state $s \in S$. All simple cycles in consistent outcomes have a sum of weights at least

[1] Note that $w_{+\varepsilon}$ can be transformed into an integer valued function without changing the answers to the considered decision problems.

equal to zero. Hence, the corresponding outcome in $G_{+\varepsilon}$ is such that all simple cycles of length n have sums at least equal to $n \cdot \varepsilon > 0$, which proves that the strategy is also winning in $G_{+\varepsilon}$.

Consider a memoryless winning strategy of $\mathcal{P}_2$ in G from initial state $s \in S$. All simple cycles in consistent outcomes have a strictly negative sum of weights, that is the sum is at most equal to -1. Hence, the corresponding outcome in $G_{+\varepsilon}$ is such that all simple cycles of length n have sums at most equal to $-1 + n \cdot \varepsilon$. Since $n \leq |S|$ and $\varepsilon < 1/|S|$, we have that the sum is strictly negative, which proves that the strategy is also winning in $G_{+\varepsilon}$.

By determinacy of mean-payoff games, we obtain the claim. □

Lemma 9.13. *For all one-dimension game $G = (S_1, S_2, E, w)$ with integer weights, for all ε, $0 < \varepsilon < 1/|S|$, for all initial state $s \in S$, if $\mathcal{P}_1$ has a strategy to win the bounded window mean-payoff problem in $G_{+\varepsilon}$, then $\mathcal{P}_1$ has a strategy to win the mean-payoff threshold problem in G.*

Proof. Assume there exists a winning strategy of $\mathcal{P}_1$ for the bounded window mean-payoff problem in $G_{+\varepsilon}$ from initial state $s \in S$. By Lemma 8.3, assertion (a), we have that this strategy ensures a non-negative mean-payoff in $G_{+\varepsilon}$. By shifting weights by $-\varepsilon$, this can be equivalently expressed as (Prop. A) the existence of a strategy of $\mathcal{P}_1$ ensuring a mean-payoff at least equal to $-\varepsilon$ in the game G.

For sufficiently small values of ε, that is for $0 < \varepsilon < 1/|S|$, we claim that (Prop. A) implies that (Prop. B) $\mathcal{P}_1$ has a strategy to ensure a non-negative mean-payoff in G. By contradiction, assume this implication is false, that is we have that (Prop. A) is true and (Prop. B) is not. It implies the following.

- *(Prop. A) is true*: $\mathcal{P}_1$ has a memoryless strategy to ensure that the mean-payoff is at least equal to $-\varepsilon$, i.e., strictly greater than $-1/|S|$.
- *(Prop. B) is false*: $\mathcal{P}_2$ has a memoryless strategy to ensure that all simple cycles in consistent outcomes have a sum of weights at most -1. Hence, this strategy ensures a mean-payoff at most equal to $-1/|S|$.

Obviously, it is not possible to have both (Prop. A) true and (Prop. B) false for any initial state $s \in S$, hence proving our claim. □

Lemma 9.14. *The one-dimension mean-payoff problem reduces in polynomial time to the bounded window mean-payoff problem.*

Proof. Let $G = (S_1, S_2, E, w)$ be a game with integer weights, and $s_{\mathsf{init}} \in S$ be the initial state. Let ε be any rational value such that $0 < \varepsilon < 1/|S|$. We claim that the answer to the mean-payoff threshold problem in G is YES if and only if the answer to the bounded window mean-payoff problem in $G_{+\varepsilon}$ is YES.

The left-to-right implication is proved in two steps. Assume the answer to the mean-payoff threshold problem in G is YES. First, by Lemma 9.12, we have that $\mathcal{P}_1$ has a strategy to ensure a strictly positive mean-payoff in $G_{+\varepsilon}$. Second, by Lemma 8.3, assertion (b), this implies that the answer to the bounded window mean-payoff problem in $G_{+\varepsilon}$ is YES.

The right-to-left implication is obtained by straightforward application of Lemma 9.13. □

Remark 9.15. The reduction established in Lemma 9.14 cannot be reversed in order to solve bounded window mean-payoff games via classical mean-payoff games. Indeed, the reduction relies on the absence of simple cycles of value zero in the game $G_{+\varepsilon}$, which is not verified in general if the reduction starts from arbitrary bounded window mean-payoff games. Indeed it does not suffice to shift the weights symmetrically by $-\varepsilon$ to obtain an equivalent mean-payoff game, as witnessed by Fig. 8.2, for which any negative shift gives a game losing for the mean-payoff threshold problem, while the bounded window problem on the original game is satisfied. ◁

9.2.3 Wrap-up

We close our study of two-player one-dimension games with Theorem 9.16.

Theorem 9.16. *In two-player one-dimension games, the bounded window mean-payoff problem is in NP ∩ coNP and at least as hard as mean-payoff games. Memoryless strategies suffice for $\mathcal{P}_1$ and infinite-memory strategies are required for $\mathcal{P}_2$ in general.*

9.3 On Direct Objectives

Through this chapter, we have studied the prefix-independent versions of the objectives defined in Sect. 8.2. In this section, we briefly argue that similar complexity results are obtained for the *direct* variants (Table 9.1), by slight modifications of the presented proofs. Notice that memory requirements however change, as it is now sufficient to force one sufficiently long (for the fixed problem) or never closing (for the bounded problem) window to make an outcome losing.

	one-dimension		
	complexity	$\mathcal{P}_1$ mem.	$\mathcal{P}_2$ mem.
direct fixed polynomial window	P-c.	mem. req. $\leq$ linear($\lvert S\rvert \cdot l_{\max}$)	
direct fixed arbitrary window	P($\lvert S\rvert, V, l_{\max}$)		
direct bounded window problem	NP $\cap$ coNP	memoryless	**linear**

Table 9.1: Complexities and memory requirements for the direct objectives in one-dimension games. Differences with the prefix-independent objectives are in bold (c. for complete).

Direct fixed window problem. The polynomial algorithm in the size of the game and the size of the window is given by Lemma 9.2. For polynomial windows, we obtain P-hardness using the proof of Lemma 9.5 and window size $l_{\max} = 2 \cdot |S|$, as if $\mathcal{P}_1$ can win the reachability game, then he has a strategy to do it in at most $|S|$ steps. Lemma 9.4 extends to direct objectives, and provides linear upper bounds on memory with the same arguments. In particular, the provided examples of games require memory for both players when the direct fixed window objective is considered.

Direct bounded window problem. We obtain an NP $\cap$ coNP algorithm for the direct bounded problem by simplifying BoundedProblem (Lemma 9.9) as follows: BoundedProblem$(G) = S \setminus$ UnbOpenWindow(G). Indeed, as the objective is no longer prefix-independent, it is sufficient for $\mathcal{P}_2$ to force one

window that never closes to make the play losing. Hence, the attractor of the set $S \setminus L$ in algorithm BoundedProblem cannot be declared winning for $\mathcal{P}_1$. While memoryless strategies still suffice for $\mathcal{P}_1$ (applying the arguments of Lemma 9.9), winning strategies for $\mathcal{P}_2$ do not need infinite memory anymore, but at most linear memory. Indeed, a winning strategy of $\mathcal{P}_2$ is the one described in the proof of Lemma 9.9, but without taking rounds into account (i.e., the play stays forever in round one). To illustrate that memoryless strategies still do not suffice for $\mathcal{P}_2$, consider a variation of Fig. 8.3, with the initial state being s_2. Clearly, $\mathcal{P}_2$ must first take the cycle to s_1 then loop forever on s_2 to ensure a never closing window. Corollary 9.10 extends in the direct case and gives the same bound on the window size. Finally, the reduction of mean-payoff games developed in Lemma 9.14 carries over to the direct bounded window objective, as the game with shifted weights is such that the mean-payoff is strictly positive. In which case, the supremum total-payoff is infinite and Lemma 8.3 applies, implying the result.

CHAPTER 10

Multi-Dimension Window Games

Fixed Window ◇ Bounded Window ◇ On Direct Objectives

For the *fixed window mean-payoff problem*, we first present an EXPTIME algorithm that computes the winning states of $\mathcal{P}_1$.

We also establish lower bounds on the complexity of the fixed window problem: the problem is EXPTIME-hard for arbitrary window sizes (both in the case of fixed weights and arbitrary dimensions, and in the case of a fixed number of dimensions and arbitrary weights), whereas it is PSPACE-hard for polynomial window sizes.

We finally show that exponential memory is both sufficient and necessary in general for both players, even for polynomial window sizes.

For the *bounded window mean-payoff problem*, we establish non-primitive-recursive-hardness.

All these results transfer to the direct variants.

These results were established in joint work with Chatterjee, Doyen and Raskin [CDRR13a, CDRR13b].

10.1 Fixed Window

Remark 10.1. Recall the definitions given in Sect. 8.2. Observe that in multi-dimension games, window objectives do not require that opened windows close simultaneously. In that sense, it is *asynchronous*.

Synchronous variants may be interesting to study but some useful properties are lost in that setting, such as the inductive property on windows. Hence our techniques cannot be extended straightforwardly. ◁

10.1.1 Algorithm

We start by providing an EXPTIME algorithm via a reduction from a fixed window mean-payoff game $G = (S_1, S_2, E, k, w)$ to an exponentially larger unweighted co-Büchi game G^c (where the objective of $\mathcal{P}_1$ is to avoid visiting a set of bad states infinitely often).

Lemma 10.2. *The fixed window mean-payoff problem over a multi-dimension game G reduces in exponential time to the co-Büchi problem on an exponentially larger game G^c.*

Recall that a winning play is such that, starting in some position $i \geq 0$, in all dimensions, all opening windows are closed in at most $l_{\max}$ steps.

We keep a counter of the sum over the sequence of edges and as soon as it turns non-negative (in at most $l_{\max}$ steps), we reset the sum counter and start a new sequence (which also must become non-negative in at most $l_{\max}$ steps). Hence, the reduction is based on accounting for each dimension the current negative sum of weights since the last reset, and the number of steps that remain to achieve a non-negative sum.

This accounting is encoded in the states of $G^c = (S_1^c, S_2^c, E^c)$, as from the original state space S, we go to $S \times (\{-l_{\max} \cdot W, \ldots, 0\} \times \{1, \ldots, l_{\max}\})^k$: states of G^c are tuples representing a state of G and the current status of open windows in all dimensions (sum and remaining steps). We add states reached whenever a window reaches its maximum size $l_{\max}$ without closing. We label those as *bad* states. We have one bad state for every state of G. Transitions in G^c are built in order to accurately model the effect of transitions of G on open windows.

Clearly, a play is winning for the fixed window problem in G if and only if the corresponding play in G^c is winning for the co-Büchi objective that asks that the set of bad states is not visited infinitely often, as that means that from some point on, all windows close in the required number of steps.

Proof. Let $G = (S_1, S_2, E, k, w)$ be a multi-dimension game with the fixed window objective $\mathsf{FixWMP}_G(\{0\}^k, l_{\max} \in \mathbb{N}_0)$ and initial state $s_{\mathsf{init}} \in S$. Recall that W denotes the maximal absolute value of any edge in E. We construct the unweighted game $G^c = (S_1^c, S_2^c, E^c)$ in the following way.

- $S_1^c = \Big(S_1 \times (\{-W \cdot l_{\max}, \dots, 0\} \times \{1, \dots, l_{\max}\})^k\Big) \cup \{\varsigma_1, \dots, \varsigma_{|S|}\}$. Additional states $\varsigma_1, \dots, \varsigma_{|S|}$ denote special *bad states*, one for each of the original states $s_1, \dots, s_{|S|} \in S$. The other states are built as tuples that represent (a) a visited state in G, (b) for each dimension, a couple modeling (b.1) the current sum of weights since the last time the sum in this dimension was non-negative, and (b.2) the number of steps that remain to reach a non-negative sum in this dimension (i.e., before reaching the maximum window size).
- $S_2^c = S_2 \times (\{-W \cdot l_{\max}, \dots, 0\} \times \{1, \dots, l_{\max}\})^k$.
- Edges $((s_a, (\sigma_a^1, \tau_a^1), \dots, (\sigma_a^k, \tau_a^k)), (s_b, (\sigma_b^1, \tau_b^1), \dots, (\sigma_b^k, \tau_b^k))$ of E^c are built as follows. For all $(s_a, s_b) \in E$, let $w_e = w((s_a, s_b))$, we have
  - $((s_a, (\sigma_a^1, \tau_a^1), \dots, (\sigma_a^k, \tau_a^k)), \varsigma_b) \in E^c$, with ς_b the bad state associated to state s_b, iff $\exists\, t$, $1 \leq t \leq k$ such that $\tau_1^t = 1$ and $\sigma_1^t + w_e(t) < 0$,
  - $((s_a, (\sigma_a^1, \tau_a^1), \dots, (\sigma_a^k, \tau_a^k)), (s_b, (\sigma_b^1, \tau_b^1), \dots, (\sigma_b^k, \tau_b^k)) \in E^c$ iff $\forall\, t$, $1 \leq t \leq k$, we have
    - $\sigma_a^t + w_e(t) \geq 0 \rightarrow \sigma_b^t = 0, \tau_b^t = l_{\max}$,
    - $\sigma_a^t + w_e(t) < 0 \wedge \tau_a^t > 1 \rightarrow \sigma_b^t = \sigma_a^t + w_e(t), \tau_b^t = \tau_a^t - 1$,

  and we add edges $(\varsigma_i, (s_i, (0, l_{\max}, \dots, (0, l_{\max}))$ to E^c for all states $s_i \in S$.

Intuitively, the game G^c is built by unfolding the game G and integrating the current sum of weights in the states of G^c, as well as the number of steps that remain to close a window, both for each dimension separately. The game G^c starts in the initial state $(s_{\mathsf{init}}, (0, l_{\max}), \dots, (0, l_{\max}))$, and each time a transition (s, s') in the original game G is taken, the game G^c is updated to a state

$(s', (\sigma^1, \tau^1), \ldots, (\sigma^k, \tau^k))$ such that (a) if the current sum becomes positive in a dimension t, the corresponding sum counter is reset to zero and the step counter is reset to its maximum value, $l_{\max}$, (b) if the sum is still strictly negative in a dimension t and the window for this dimension is not at its maximal size, the sum is updated and the step counter is decreased, and (c) if the sum stays strictly negative and the maximal size is reached in any dimension, the game visits the corresponding bad state and then, all counters are reset for all dimensions.

We argue that a play π in G is winning for the fixed window mean-payoff objective if and only if the corresponding play π^c in G^c is winning for the co-Büchi objective asking not to visit the set $S_\varsigma = \{\varsigma_1, \ldots, \varsigma_{|S|}\}$ infinitely often.

Indeed, consider a play π that is winning for objective $\mathsf{FixWMP}_G(\{0\}^k, l_{\max})$. By eq. (8.4), this play only sees a finite number of bad windows (windows that are not closed in $l_{\max}$ steps in some dimension). By construction of G^c, the corresponding play π^c only visits the set S_ς a finite number of times, hence it is winning for the co-Büchi objective. Now, let π^c be a winning play for the co-Büchi objective. By definition, there exists a position i in π^c such that all states appearing after position i belong to $S \setminus S_\varsigma$. It remains to prove that for any position $j \geq i$, for any dimension t, $1 \leq t \leq k$, there is a valid window of size at most $l_{\max}$. Again we use the inductive property of windows. We know by construction that a reset of the sum happens in at most $l_{\max}$ steps, otherwise we go to a bad state. Assume j is a position with a sum counter of zero in some dimension t, and j' is the next such position. Since resets are done *as soon as* the sum becomes non-negative, all suffixes of the sequence from j to j' are non-negative. Hence, it is clear that for all position j'', $j < j'' < j'$, the window from j'' to j' in dimension t is closed. Consequently, the corresponding play π in G is winning for the fixed window mean-payoff objective of threshold 0 and window size $l_{\max}$. □

As a direct corollary of this reduction, we obtain an EXPTIME algorithm to solve the fixed window mean-payoff problem on multi-dimension games, as solving co-Büchi games takes quadratic time in the size of the game [CH12].

Corollary 10.3. *Given a two-player multi-dimension game $G = (S_1, S_2, E, k, w)$ and a window size $l_{\max} \in \mathbb{N}_0$, the fixed window mean-payoff problem can be solved in time $\mathcal{O}(|S|^2 \cdot (l_{\max})^{4 \cdot k} \cdot W^{2 \cdot k})$ via a reduction to co-Büchi games.*

Proof. Lemma 10.2 uses a co-Büchi game which state space is of size

$$\left|S \times \left(\{-W \cdot l_{\max}, \ldots, 0\} \times \{1, \ldots, l_{\max}\}\right)^k\right| + |S| = \mathcal{O}\Big(|S| \cdot (l_{\max})^{2 \cdot k} \cdot W^k\Big).$$

The quadratic algorithm for co-Büchi games described in [CH12] implies the result. □

A natural question is whether a distinct algorithm is useful in the one-dimension case. Remark 10.4 notes that it is.

Remark 10.4. The algorithm described in Corollary 10.3 yields a procedure which is polynomial in the size of the state space, the window size, and the largest weight for the subclass of one-dimension games, hence only *pseudo-polynomial* (i.e., exponential in V, the length of the encoding of weights), whereas Lemma 9.3 gives an algorithm that is also polynomial in the encoding of weights. ◁

10.1.2 Memory and Complexity Bounds

We first consider the fixed *arbitrary* window problem for which we show (i) in Lemma 10.5, EXPTIME-hardness for $\{-1, 0, 1\}$ weights and arbitrary dimensions via a reduction from the *membership problem for alternating polynomial-space Turing machines (APTMs)* [CKS81], and (ii) in Lemma 10.6, EXPTIME-hardness for two dimensions and arbitrary weights via *countdown games* [JSL08].

Arbitrary windows - membership problem in APTMs. Let $\mathcal{A}$ be an APTM and $\zeta \in \{0, 1\}^*$ a word, such that the tape contains at most $p(|\zeta|)$ cells, where p is a polynomial function. The membership problem asks to decide if $\mathcal{A}$ accepts ζ.

We build a fixed arbitrary window mean-payoff game G so that $\mathcal{P}_1$ has to simulate the run of $\mathcal{A}$ on ζ, and $\mathcal{P}_1$ has a winning strategy in G if and only if the word is accepted by the machine. For each tape cell $h \in \{1, 2, \ldots, p(|\zeta|)\}$, we have two dimensions, $(h, 0)$ and $(h, 1)$ such that a sum of weights of value -1 (i.e., an open window) in dimension (h, i), $i \in \{0, 1\}$ encodes that in the current configuration of $\mathcal{A}$, tape cell h contains a bit of value i.

In each step of the simulation (Fig. 10.1), $\mathcal{P}_1$ has to disclose the symbol under the tape head: if in position h, $\mathcal{P}_1$ discloses a 0 (resp. a 1), he obtains a reward 1 in dimension $(h, 0)$ (resp. $(h, 1)$).

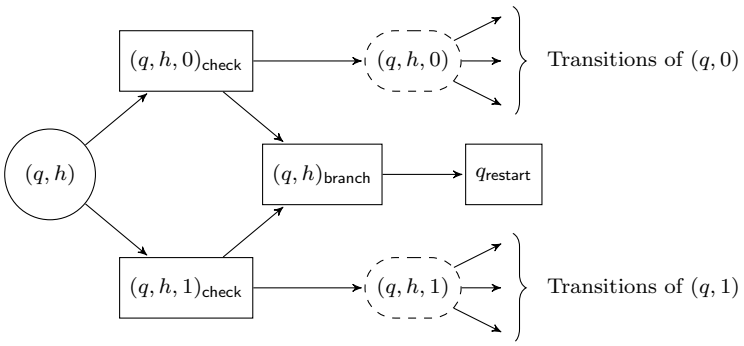


Figure 10.1: Gadget ensuring a correct simulation of the APTM on tape cell h.

To ensure that $\mathcal{P}_1$ was faithful, $\mathcal{P}_2$ is then given the choice to either let the simulation continue, or assign a reward 1 in all dimensions except $(h,0)$ and $(h,1)$ and then restart the game after looping in a zero self-loop for an arbitrary long time. If $\mathcal{P}_1$ cheats by not disclosing the correct symbol under tape cell h, $\mathcal{P}_2$ can punish him by branching to the restart state and ensuring a sufficiently long open window in the corresponding dimension before restarting (as in Fig. 8.3). But if $\mathcal{P}_1$ discloses the correct symbol and $\mathcal{P}_2$ still branches, all windows close.

In the accepting state, all windows are closed and the game is restarted. The window size $l_{\max}$ of the game is function of the existing bound on the length of an accepting run. To force $\mathcal{P}_1$ to go to the accepting state, we add an additional dimension, with weight -1 on the initial edge of the game and weight 1 on reaching the accepting state.

Lemma 10.5. *The fixed arbitrary window mean-payoff problem is EXPTIME-hard in multi-dimension games with* $\{-1,0,1\}$ *weights and arbitrary dimensions.*

Proof. An *alternating Turing machine* (ATM) [CKS81] is written as a tuple $\mathcal{A} = (Q, q_0, \Sigma_{\mathsf{in}}, \delta, q_{\mathsf{acc}})$ where:

- Q is the finite set of control states with a partition $(Q_\vee, Q_\wedge)$ of Q into existential and universal states;
- $q_0 \in Q$ is the initial state;
- $\Sigma_{\mathsf{in}} = \{0,1\}$ is the input alphabet and $\Sigma_{\mathsf{tape}} = \Sigma_{\mathsf{in}} \cup \{\#\}$ the tape alphabet, with $\#$ the blank symbol;

- $\delta \subseteq Q \times \Sigma_{\mathsf{tape}} \times Q \times \Sigma_{\mathsf{tape}} \times \{-1, 1\}$ is a transition relation;
- there is a special accepting state $q_{\mathsf{acc}} \in Q_\vee$ (without loss of generality).

We say that $\mathcal{A}$ is a *polynomial-space* alternating Turing machine (APTM) if for some polynomial function p, the space used by $\mathcal{A}$ on any input word $\zeta \in \Sigma_{\mathsf{in}}^*$ is bounded by $p(|\zeta|)$.

We define the AND-OR graph of the APTM $(\mathcal{A}, p)$ on the input word $\zeta \in \Sigma_{\mathsf{in}}^*$ as $\mathcal{G}(\mathcal{A}, p) = \langle S_\vee, S_\wedge, s_0, \Delta, R\rangle$ where

- $S_\vee = \{(q, h, t) \mid q \in Q_\vee,\ 1 \leq h \leq p(|\zeta|) \text{ and } t \in \Sigma_{\mathsf{tape}}^{p(|\zeta|)}\}$;
- $S_\wedge = \{(q, h, t) \mid q \in Q_\wedge,\ 1 \leq h \leq p(|\zeta|) \text{ and } t \in \Sigma_{\mathsf{tape}}^{p(|\zeta|)}\}$;
- $s_0 = (q_0, 1, t)$ where $t = \zeta \cdot \#^{p(|\zeta|)-|\zeta|}$;
- $((q_1, h_1, t_1), (q_2, h_2, t_2)) \in \Delta$ iff there exists $(q_1, t_1(h_1), q, \gamma, d) \in \delta$ such that $q_2 = q$, $h_2 = h_1 + d$, $t_2(h_1) = \gamma$ and $t_2(h) = t_1(h)$ for all $h \neq h_1$;
- $R = \{(q, h, t) \in S_\vee \mid q = q_{\mathsf{acc}}\}$.

Intuitively, states of the graph correspond to configurations (q, h, t) where q is a control state of the machine, h the position of the tape head, and t the current word written on the tape. Given a state q of the machine $\mathcal{A}$, tape head on cell h and a word t on the tape, a transition from (q, h, t) to (q', h', t') exists in the graph $\mathcal{G}(\mathcal{A}, p)$ if the transition relation δ of the machine $\mathcal{A}$ admits a transition that given this configuration, updates the content of cell h to the symbol $t'(h)$, such that the tape now contains the word t', and then goes to control state q' and moves the tape head to an adjacent cell h'.

A word $\zeta \in \Sigma_{\mathsf{in}}^*$ is *accepted* by an APTM $(\mathcal{A}, p)$ if there exists a run tree (obtained by choosing a child in existential nodes and keeping all children in universal nodes) of $\mathcal{A}$ on ζ such that all leafs are accepting configurations. That is, a word is accepted if and only if, in the two-player game defined by $\mathcal{G}(\mathcal{A}, p)$, player $\mathcal{P}_\vee$ has a strategy to reach the set of accepting states R. Deciding the acceptance of a word by an APTM is an EXPTIME-complete problem, known as the membership problem [CKS81].

We construct a fixed window mean-payoff game $G = (S_1, S_2, E, k, w)$ simulating the machine $(\mathcal{A}, p)$ as follows. Let $k = 2 \cdot p(|\zeta|) + 1$: there is a dimension for

each pair $(h,0)$ and $(h,1)$, for all $1 \leq h \leq p(|\zeta|)$, and one additional dimension. The set of states S of the game is

$$\begin{aligned} S = &\{q_{\mathsf{restart}}\} \cup \{q_{\mathsf{in}}\} \cup \{\widehat{q_{\mathsf{acc}}}\} \\ &\cup \{(q,h) \mid q \in Q,\ 1 \leq h \leq p(|\zeta|)\} \\ &\cup \{(q,h,i)_{\mathsf{check}} \mid q \in Q,\ 1 \leq h \leq p(|\zeta|), i \in \{0,1\}\} \\ &\cup \{(q,h)_{\mathsf{branch}} \mid q \in Q,\ 1 \leq h \leq p(|\zeta|)\} \\ &\cup \{(q,h,i) \mid q \in Q,\ 1 \leq h \leq p(|\zeta|), i \in \{0,1\}\}. \end{aligned}$$

States of the form (q,h) belong to $\mathcal{P}_1$. States of the form (q,h,i) belong to $\mathcal{P}_1$ if $q \in Q_\vee$ in the machine $\mathcal{A}$. All other states belong to $\mathcal{P}_2$. The initial state is q_{restart}. It has two outgoing edges with weights zero in all dimensions: one self-loop, and one edge to q_{in}. The latter is assigned the following weights: -1 for dimension (h,i) if the letter at position h of ζ is i, -1 in the very last dimension $(2 \cdot p(|\zeta|)+1)$, and zero everywhere else. From q_{in}, the game goes to $(q_0,1)$ and the simulation of $\mathcal{A}$ begins.

The game mimics runs of $\mathcal{A}$, and it is ensured that if the current state of the game is (q,h) and the cell content is i, then the sum of weights since the last visit of q_{in} in dimension (h,i) is -1. We refer to the segment of play since the last visit of q_{in} as the *current round*. We depict a step of the simulation in Fig. 10.1. At state (q,h), $\mathcal{P}_1$ has the choice between states $(q,h,0)_{\mathsf{check}}$ and $(q,h,1)_{\mathsf{check}}$, resp. corresponding to declaring a content 0 or 1 of the tape cell h. The reward for dimension (h,i), $i \in \{0,1\}$ is 1 on state $(q,h,i)_{\mathsf{check}}$. At state $(q,h,i)_{\mathsf{check}}$, a state of $\mathcal{P}_2$, $\mathcal{P}_2$ checks whether $\mathcal{P}_1$ has correctly revealed the tape content as follows: (i) Player $\mathcal{P}_2$ can choose to go to state $(q,h)_{\mathsf{branch}}$, in which all dimensions other than $(h,0)$ and $(h,1)$, including the very last, are increased by 1, and then go to q_{restart} on which $\mathcal{P}_2$ will be able to delay the play; (ii) Player $\mathcal{P}_2$ can choose to proceed and continue the simulation: the game then goes to state (q,h,i). State (q,h,i) is either a state of $\mathcal{P}_1$ or $\mathcal{P}_2$, depending on the affiliation of state q in the APTM. Such a gadget ensures that if $\mathcal{P}_1$ cheats by not disclosing the correct symbol, $\mathcal{P}_2$ can force an open window of arbitrary length in the current round by looping on q_{restart} for some time, and then restarting the game. On the other hand, if $\mathcal{P}_1$ is faithful and $\mathcal{P}_2$ still decides to branch to $(q,h)_{\mathsf{branch}}$, then all windows will be closed for the current round.

If $\mathcal{P}_1$ does not cheat and $\mathcal{P}_2$ acknowledges it by not branching, the game advances to a state of the form (q, h, i). At such a state, we add transitions as follows: if there exists a transition from (q, h, i) to (q', h', i') in $\mathcal{A}$, then we add an edge from (q, h, i) to (q', h') in the game G, and assign weight -1 in dimension (h, i'), as the tape cell at position h contains i' and we ensure that the sum in dimension (h, i') in the current round is -1. At the accepting states (q_{acc}, h), all dimensions are assigned reward 1, and the next state is $\widehat{q_{\mathsf{acc}}}$. State $\widehat{q_{\mathsf{acc}}}$ is followed by q_{restart}. Again there is no risk in looping as all dimensions are now non-negative.

Formally, blank symbols need to be added. For brevity and simplicity of the presentation, we omit these technical details.

We fix the window size $l_{\max}$ equal to three times the size of the configuration graph (bound on the length of a run) plus three, and we argue that the game G is a faithful simulation of the machine $\mathcal{A}$, that is, $\mathcal{P}_1$ wins the fixed window mean-payoff game if and only if the word ζ is accepted by $\mathcal{A}$. Notice that the construction ensures that if $\mathcal{P}_1$ cheats in the current round, $\mathcal{P}_2$ can make this round losing, as discussed before. Similarly, if $\mathcal{P}_1$ does not cheat but does not reach the accepting state, dimension $2 \cdot p(|\zeta|) + 1$ will remain negative when arriving in q_{restart} and $\mathcal{P}_2$ will be able to cycle long enough to make the round losing as the window in the last dimension will remain open for $l_{\max}$ steps. Clearly, $\mathcal{P}_1$ cannot see losing rounds infinitely often otherwise the play is losing.

Assume the word ζ is accepted by the machine. Then there is an accepting run tree, and the winning strategy of $\mathcal{P}_1$ is to follow this run tree and always reveal the correct symbol. This way, either $\mathcal{P}_2$ restarts and the round is winning because all dimensions are non-negative, or $\mathcal{P}_2$ does not restart and an accepting state (q_{acc}, h) is reached within the maximum allowed window size. Indeed, in the APTM, there is a strategy to reach the accepting state in a number of steps bounded by the size of the configuration graph. In that case, the round is also winning.

Conversely, assume that the word ζ is not accepted by the APTM. Consider any strategy λ_1 of $\mathcal{P}_1$. Clearly, $\mathcal{P}_1$ cannot cheat as otherwise, he loses. So assume he does not cheat. Then there is a path in the run tree obtained from playing the strategy λ_1 in $\mathcal{A}$ such that the path never reaches an accepting state. Hence, the strategy λ_2 of $\mathcal{P}_2$ that follows this path in the game G ensures that the sum

in dimension $2 \cdot p(|\zeta|) + 1$ is always strictly negative, and after waiting till the bound $l_{\max}$ on the window size is met, $\mathcal{P}_2$ has made the round losing and he can restart the game safely. Acting this way infinitely often, $\mathcal{P}_2$ can violate the fixed window objective for $\mathcal{P}_1$. It follows that $\mathcal{P}_1$ wins in G if and only if the word ζ is accepted by the APTM $\mathcal{A}$. □

Arbitrary windows - countdown games. We now prove that the problem is also EXPTIME-hard for two dimensions and arbitrary weights via a reduction from countdown games. A countdown game $\mathcal{C}$ consists of a weighted graph $(\mathcal{S}, \mathcal{T})$, with $\mathcal{S}$ the set of states and $\mathcal{T} \subseteq \mathcal{S} \times \mathbb{N}_0 \times \mathcal{S}$ the transition relation. Configurations are of the form (s, c), $s \in \mathcal{S}$, $c \in \mathbb{N}$. The game starts in an initial configuration (s_{init}, c_0) and transitions from a configuration (s, c) are performed as follows: first $\mathcal{P}_1$ chooses a duration d, $0 < d \leq c$ such that there exists $t = (s, d, s') \in \mathcal{T}$ for some $s' \in \mathcal{S}$, second $\mathcal{P}_2$ chooses a state $s' \in \mathcal{S}$ such that $t = (s, d, s') \in \mathcal{T}$. Then, the game advances to $(s', c - d)$. Terminal configurations are reached whenever no legitimate move is available. If such a configuration is of the form $(s, 0)$, $\mathcal{P}_1$ wins the play. Otherwise, $\mathcal{P}_2$ wins the play. Deciding the winner in countdown games given an initial configuration (s_{init}, c_0) is EXPTIME-complete [JSL08].

Given a countdown game $\mathcal{C}$ and an initial configuration (s_{init}, c_0), we create a game $G = (S_1, S_2, E, k, w)$ with $k = 2$ and a fixed window objective for $l_{\max} = 2 \cdot c_0 + 2$ (Fig. 10.2). The two dimensions are used to store the value of the countdown counter and its opposite. Each time a duration d is chosen, an edge of value $(-d, d)$ is taken. The game simulates the moves available in $\mathcal{C}$: a strict alternation between states of $\mathcal{P}_1$ (representing states of $\mathcal{S}$) and states of $\mathcal{P}_2$ (representing transitions available from a state of $\mathcal{S}$ once a duration has been chosen). On states of $\mathcal{P}_1$, we add the possibility to branch to a state s_{restart} of $\mathcal{P}_2$, in which $\mathcal{P}_2$ can either take a zero cycle, or go back to the initial state and force a restart of the game. By placing weights $(0, -c_0)$ on the initial edge, and $(c_0, 0)$ on the edge branching to s_{restart}, we ensure that the only way to win for $\mathcal{P}_1$ is to accumulate a value exactly equal to c_0 in the game before switching to s_{restart}. This is possible if and only if $\mathcal{P}_1$ can reach a configuration of value zero in $\mathcal{C}$.

Lemma 10.6. *The fixed arbitrary window mean-payoff problem is EXPTIME-hard in games with two dimensions and arbitrary weights.*

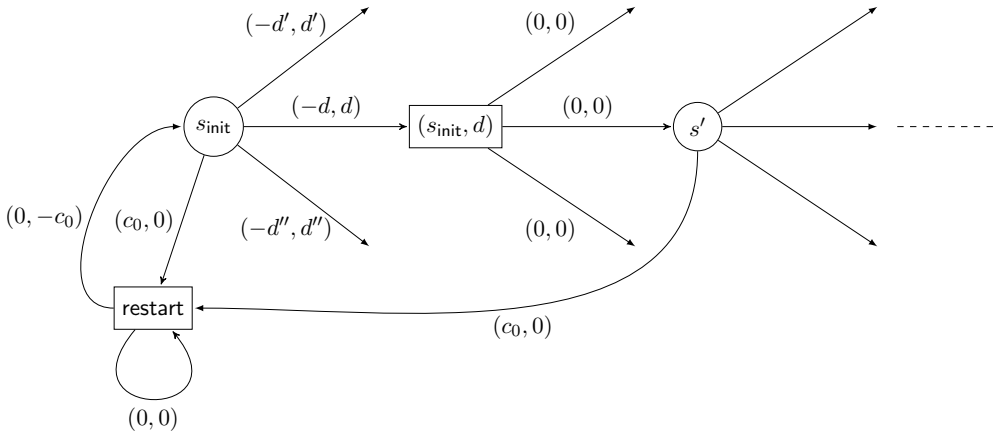


Figure 10.2: Reduction from countdown games to fixed window games.

Proof. We establish a polynomial-time reduction from the countdown game problem to the fixed arbitrary window problem. Let $\mathcal{C} = (\mathcal{S}, \mathcal{T})$ be a countdown game [JSL08], with initial configuration (s_{init}, c_0). We create a corresponding game $G = (S_1, S_2, E, k, w)$ as follows.

- $S_1 = \mathcal{S}$.
- Let $S^{\mathcal{T}} \subseteq \mathcal{S} \times \mathbb{N}_0$ be the subset of pairs (s, d) such that there exists a transition $(s, d, s') \in \mathcal{T}$. Then, $S_2 = S^{\mathcal{T}} \cup \{s_{\mathsf{restart}}\}$. State s_{restart} is the initial state of game G.
- For each transition $(s, d, s') \in \mathcal{T}$, we add edges $(s, (s, d))$, with $s \in S_1$ and $(s, d) \in S_2$, and $((s, d), s')$, with $s' \in S_1$, to the set of edges E. Edge $(s, (s, d))$ has weight $(-d, d)$ and edge $((s, d), s')$ has weight $(0, 0)$.
- For all $s \in S_1$, we add an edge $(s, s_{\mathsf{restart}})$ of weight $(c_0, 0)$.
- From s_{restart}, we add an edge $(s_{\mathsf{restart}}, s_{\mathsf{init}})$ of value $(0, -c_0)$.
- On s_{restart}, we add a self-loop $(s_{\mathsf{restart}}, s_{\mathsf{restart}})$ of weight $(0, 0)$.

We fix the window size $l_{\max} = 2 \cdot c_0 + 2$, and we claim that $\mathcal{P}_1$ wins the fixed window problem if and only if he wins the countdown game. Recall that to win a countdown game, $\mathcal{P}_1$ must be able to reach a configuration $(s, 0)$ in the game $\mathcal{C}$. The key idea to our construction is that in the game G, the only way to avoid

seeing infinitely often open windows of size larger than $l_{\max}$ is to accumulate exactly c_0 before restarting, which is equivalent to reaching a configuration of value 0 in $\mathcal{C}$.

Notice that the game G starts by visiting an edge of value $(0, -c_0)$ and afterwards, all edges from states of $\mathcal{P}_1$ have a value $(-d, d)$ corresponding to the duration he chooses in the countdown game. All except the edge he can decide to take to go to s_{restart}, which value is $(c_0, 0)$. Clearly, if $\mathcal{P}_1$ decides to go in s_{restart}, he has to close all windows, as otherwise $\mathcal{P}_2$ can use the self-loop to delay the play long enough and provoke a sufficiently long bad window, which if done repeatedly, induces a losing play. On the other hand, if $\mathcal{P}_1$ decides to never go toward s_{restart}, he will keep accumulating negative values in the first dimension and he is guaranteed to lose. So obviously the behavior of $\mathcal{P}_1$ should be to play as in the countdown game to accumulate exactly c_0 in dimension 2 (and $-c_0$ in dimension 1) before switching to s_{restart}, so that $\mathcal{P}_2$ can do no harm by delaying the play as all windows will be closed.

The accumulated value has to be *exactly* c_0 as (a) if it is less than c_0, dimension 2 will remain negative, and (b) if it is more than c_0, dimension 1 will stay negative (i.e., the edge $(s, s_{\mathsf{restart}})$ will not suffice to get it back above zero). Since the minimal increase is of 1 every two edges by construction, the allowed window size $l_{\max}$ is sufficient to enforce such a behavior, if possible. This shows that $\mathcal{P}_1$ wins the fixed window problem from initial state s_{restart} in G if and only if he also wins the countdown game $\mathcal{C}$ from (s_{init}, c_0), as accumulating c_0 in G is equivalent to reaching a configuration of value zero in $\mathcal{C}$. □

Polynomial windows - generalized reachability games. For the case of polynomial windows, Lemma 10.7 proves PSPACE-hardness via a reduction from generalized reachability games [FH10]. Filling the gap with the EXPTIME-membership given by Corollary 10.3 is an open problem.

The generalized reachability objective is a conjunction of reachability objectives: a winning play has to visit a state of each of a series of k reachability sets. If $\mathcal{P}_1$ has a winning strategy in a generalized reachability game $G^r = (S_1^r, S_2^r, E^r)$, then he has one that guarantees visit of all sets within $k \cdot |S^r|$ steps.

We create a modified weighted version of the game, $G = (S_1, S_2, E, k, w)$, such that the weights are k-dimension vectors. The game starts by opening a window in all dimensions and the only way for $\mathcal{P}_1$ to close the window in

dimension t, $1 \leq t \leq k$ is to reach a state of the t-th reachability set. We modify the game by giving $\mathcal{P}_2$ the ability to close all open windows and restart the game such that the prefix-independence of the fixed window objective cannot help $\mathcal{P}_1$ to win without reaching the target sets. Then, a play is winning in G for the fixed window objective of size $l_{\max} = 2 \cdot k \cdot |S^r|$ if and only if it is winning for the generalized reachability objective in G^r.

Lemma 10.7. *The fixed polynomial window mean-payoff problem is PSPACE-hard.*

Proof. We show the PSPACE-hardness by a reduction from the generalized reachability problem [FH10]. Given a game graph $G^r = (S_1^r, S_2^r, E^r)$, a series of reachability sets $R_t \subseteq S^r$, for $1 \leq t \leq k$, with $k \leq |S^r|$, and an initial state $s^r_{\mathsf{init}} \in S^r$, the generalized reachability problem asks if there exists a strategy of $\mathcal{P}_1$ such that any consistent outcome starting in s^r_{init} visits a state of each set R_t at least once. It is known that if such a strategy exists, then there exists one which ensures reaching all sets in at most $k \cdot |S^r|$ steps.

We build a k-dimension fixed window mean-payoff game $G = (S_1, S_2, E, k, w)$ as follows. We define $S_{\mathsf{branch}} \subset S_2$, the set of $\mathcal{P}_2$ states such that for all $s, s' \in S^r$ such that $(s, s') \in E^r$, we have that $b_{s,s'} \in S_{\mathsf{branch}}$. Let $S_1 = S_1^r$ and $S_2 = S_2^r \cup S_{\mathsf{branch}} \cup \{s_{\mathsf{restart}}\}$. Let E be the set of edges such that for all $(s, s') \in E^r$, we have that $(s, b_{s,s'}) \in E$, $(b_{s,s'}, s') \in E$, $(b_{s,s'}, s_{\mathsf{restart}}) \in E$, and such that $(s_{\mathsf{restart}}, s^r_{\mathsf{init}}) \in E$. That is, we introduce in all edges of E^r a state of $\mathcal{P}_2$ that let him branch to an added state s_{restart} or continue as in G^r. The new initial state in G is s_{restart}, and there is an edge from s_{restart} to the old initial state s^r_{init}. The weights are as follows: all edges from states $b_{s,s'}$ to s_{restart} have value 1 in all dimensions. The edge from s_{restart} to s^r_{init} has value -1 in all dimensions. All other edges of the game have value zero, except edges entering a state that belongs to a reachability set R_t, which have value 1 in dimension t and 0 in the other dimensions. If a state belongs to several sets, then all corresponding dimensions get a 1.

We claim that $\mathcal{P}_1$ has a winning strategy for $\mathsf{FixWMP}_G(\{0\}^k, l_{\max} = 2 \cdot k \cdot |S^r|)$ if and only if he has a winning strategy for the generalized reachability objective in G^r. Consider the game G. Clearly, the only edge involving negative values is $(s_{\mathrm{restart}}, s^r_{\mathsf{init}})$, which value is $(-1, \ldots, -1)$. Therefore, a losing play for eq. (8.4) should see this edge infinitely often, as it is the starting position of all open

windows. On the other hand, going from a state $b_{s,s'}$ to s_{restart} involves an edge of value $(1,\dots,1)$, hence if the open window starting in s_{restart} comes back in s_{restart} before hitting its maximal size, the window will close. So the strategy of $\mathcal{P}_2$ should be to wait for $l_{\max} = 2{\cdot}k{\cdot}|S^r|$ steps before forcing a restart. Consider a winning strategy λ_1 of $\mathcal{P}_1$ in G. Because of the strategy of $\mathcal{P}_2$, λ_1 has to ensure obtaining $+1$ in all dimensions by only using transitions entering in states of S^r. By construction, this implies that λ_1 enforces a visit of all reachability sets, and thus wins for the generalized reachability problem. Consider the converse implication. Let λ_1^r be a winning strategy in G^r. There exists such a strategy that ensures seeing all reachability sets (thus closing all windows) in at most $l_{\max} = 2 \cdot k \cdot |S^r|$ steps if $\mathcal{P}_2$ does not branch to s_{restart}. On the other hand, if $\mathcal{P}_2$ does branch before $l_{\max}$ steps, all windows also close, as branching edges have value $(1,\dots,1)$. Hence, this strategy is also winning for $\mathsf{FixWMP}_G(\{0\}^k, l_{\max})$. This shows the correctness of the reduction and concludes our proof. □

Memory bounds. We conclude our study of the multi-dimension fixed window problem by considering memory bounds. A direct corollary of Lemma 10.2 is the existence of winning strategies of at most exponential size for both players, as memoryless strategies are sufficient in co-Büchi games [EJ91]. A corollary of the reduction from generalized reachability games to the fixed polynomial window problem used to prove Lemma 10.7 and the results of [FH10, Lemma 2] (showing exponential lower bounds on memory for generalized reachability objectives) is that such memory is needed in general, again for both players.

Another example of a family of games in which $\mathcal{P}_1$ requires exponential memory (in the number of dimensions) is given by the family defined in Lemma 5.9 (Fig. 5.4), introduced in the context of multi energy games. All examples have in common that the players must be able to differentiate between an exponential number of histories and act accordingly to achieve their objective: in the game of Fig. 5.4, $\mathcal{P}_1$ wins objective $\mathsf{FixWMP}_G(\{0\}^k, l_{\max} = |S|/2)$ only if he is able to make in t_i the opposite choice of $\mathcal{P}_2$ in s_i, which requires a strategy encoded as a Moore machine with at least $2^{k/2}$ states. Lemma 10.8 sums up these results.

Lemma 10.8. *In multi-dimension games with a fixed window mean-payoff objective, exponential memory is both sufficient and necessary for both players in general, even for polynomial window sizes.*

10.1.3 Wrap-up

We summarize the complexity of the fixed window problem in Theorem 10.9.

Theorem 10.9. *In two-player multi-dimension games, the fixed arbitrary window mean-payoff problem is EXPTIME-complete, and the fixed polynomial window mean-payoff problem is PSPACE-hard. For both players, exponential memory is sufficient and is required in general.*

10.2 Bounded Window

Unlike the one-dimension case, in which it is easier to decide the bounded problem than the fixed arbitrary one (i.e., the problem becomes easier when the fixed window size is sufficiently large), we prove that the complexity of the bounded window problem in multi-dimension games is at least non-primitive recursive.[1] Hence, there is no hope for efficient algorithms on the complete class of two-player multi-dimension games.

This result is obtained through a reduction from the problem of deciding the existence of an infinite execution in a *marked reset net*, also known as the *termination problem*. A marked reset net [DFS98] is a Petri net [Esp96] with *reset arcs* together with an initial marking of its places. Reset arcs are special arcs that reset a place (i.e., empty it of all its tokens). The termination problem for reset nets is decidable but non-primitive-recursive-hard (as follows from the results of [Sch02], also discussed in [LNO+08]).

Given a reset net $\mathcal{N}$ with an initial marking $\overline{m_0} \in \mathbb{N}^{|P|}$ (where P is the set of places of the net), we build a two-player multi-dimension game G with $k = |P|+3$ dimensions such that $\mathcal{P}_1$ wins the bounded window objective for threshold $\{0\}^k$ if and only if $\mathcal{N}$ does not have an infinite execution from $\overline{m_0}$.

A high level description of our reduction is as follows. The structure of the game (Fig. 10.3) is based on the alternance between two gadgets simulating the net (Fig. 10.4). Edges are labeled by k-dimension weight vectors such that the first $|P|$ dimensions are used to encode the number of tokens in each place.

[1] That is, there exists no primitive recursive function that computes the answer to the bounded window problem. A well-known example of a decidable but non-primitive recursive function is the Ackermann function [Ack28].

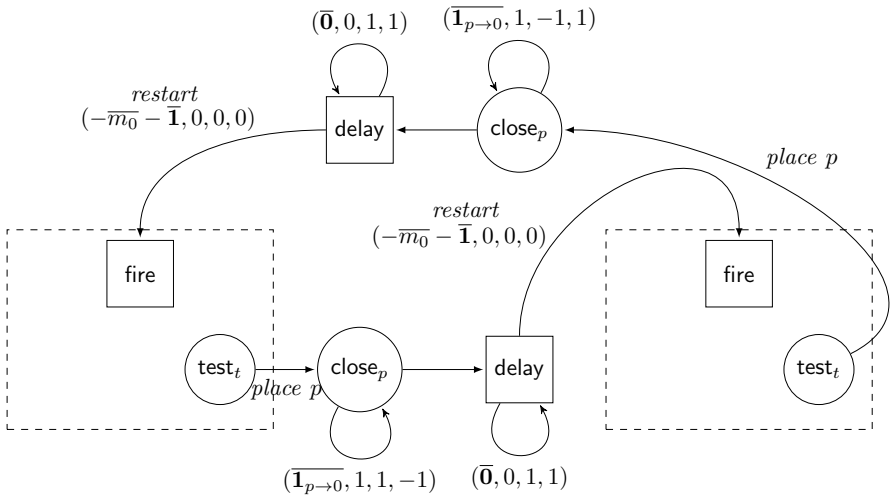


Figure 10.3: Careful alternation between gadgets is needed for $\mathcal{P}_1$ to win.

In each gadget, $\mathcal{P}_2$ chooses transitions to simulate an execution of the net. During a faithful simulation, there is always a running open window in all the first $|P|$ dimensions: if place p contains n tokens then the negative sum from the start of the simulation is $-(n+1)$. This is achieved as follows: if a transition t consumes $\mathbf{I}(t)(p)$ tokens from p, then this value is added on the corresponding dimension, and if t produces $\mathbf{O}(t)(p)$ tokens in p, then $\mathbf{O}(t)(p)$ is removed from the corresponding dimension. When a place p is reset, a gadget ensures that dimension p reaches value -1 (the coding of zero tokens). This is thanks to the monotonicity property of reset nets: if $\mathcal{P}_1$ does not simulate a full reset, then the situation gets easier for $\mathcal{P}_2$ as it leaves him more tokens available.

If all executions terminate, $\mathcal{P}_2$ has to choose an unfireable transition at some point, consuming unavailable tokens from some place $p \in P$. If so, the window in dimension p closes. After each transition choice of $\mathcal{P}_2$, $\mathcal{P}_1$ can either continue the simulation or branch out of the gadget to close all windows, except in some dimension p of his choice. Then $\mathcal{P}_2$ can arbitrarily extend any still open window in the first $(|P|+1)$ dimensions and restart the game afterwards. Dimension $(|P|+1)$ prevents $\mathcal{P}_1$ from staying forever in a gadget.

If an infinite execution exists, $\mathcal{P}_2$ simulates it and never has to choose an unfireable transition. Hence, when $\mathcal{P}_1$ branches out, the window in some di-

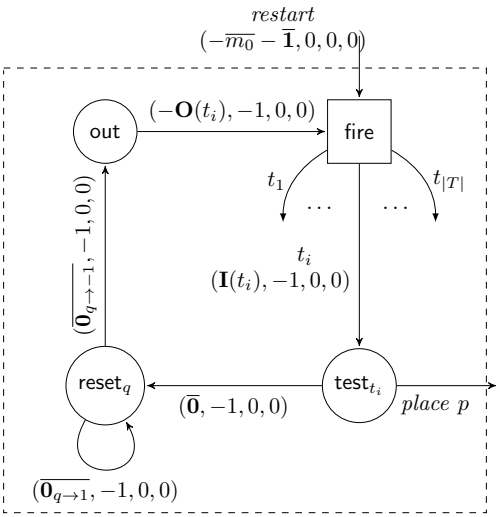


Figure 10.4: Gadget simulating an execution of the reset net.

mension p stays open. The last two dimensions force him to alternate between gadgets so that he cannot take profit of the prefix-independence to win after a faithful simulation. So, $\mathcal{P}_2$ can delay the closing of the open window for longer and longer, thus winning the game.

Theorem 10.10. *In two-player multi-dimension games, the bounded window mean-payoff problem is non-primitive-recursive-hard.*

Proof. We prove a reduction from the termination problem on reset nets to the bounded window problem on two-player multi-dimension games. The former is known to be non-primitive-recursive-hard [Sch02, LNO+08].

Let $\mathcal{N} = \langle P, T, \mathbf{I}, \mathbf{O}, r\rangle$ be a *reset net* such that

- $P = \{p_1, p_2, \ldots, p_{|P|}\}$ is the set of places;
- $T = \{t_1, t_2, \ldots, t_{|T|}\}$ is the set of transitions;
- $\mathbf{I}\colon T \to \mathbb{N}^{|P|}$ is the input function, such that for each transition $t \in T$, $\mathbf{I}(t)$ is a $|P|$-dimension vector such that for all dimension $p \in \{1, \ldots, |P|\}$, $\mathbf{I}(t)(p)$ specifies the number of tokens from place p consumed by the transition t;[2]

[2]For simplicity, we use p to refer to a place $p \in P$ and to the number $i \in \{1, \ldots, |P|\}$ such that $p_i = p$, that is p indistinctly refers to the place and the corresponding dimension in the weight vectors.

- $\mathbf{O}\colon T \to \mathbb{N}^{|P|}$ is the output function, such that for each $t \in T$, $\mathbf{O}(t)$ is a $|P|$-dimension vector such that for all dimension $p \in \{1, \ldots, |P|\}$, $\mathbf{O}(t)(p)$ specifies the number of tokens produced in place p by the transition t;
- $r\colon T \to P$ is the reset function, such that for all transition $t \in T$, $r(t)$ specifies the unique place (w.l.o.g.) which is reset by transition t.

Given an initial marking of the places (i.e., an initial number of tokens in each place) $\overline{m_0} \in \mathbb{N}^{|P|}$, the termination problem asks if there exists an infinite execution of the net, that is, if there exists an infinite sequence of transitions that can be fired from $\overline{m_0}$. A transition t is *fireable* from marking $\overline{m} \in \mathbb{N}^{|P|}$ if for all place $p \in P$, $\mathbf{I}(t)(p) \leq \overline{m}(p)$. An execution terminates if no transition can be fired because the necessary tokens are unavailable. We first note an important *monotonicity* property of reset nets: for all reset net $\mathcal{N} = \langle P, T, \mathbf{I}, \mathbf{O}, r\rangle$, for all markings $\overline{m}, \overline{n} \in \mathbb{N}^{|P|}$, if $\overline{m} \leq \overline{n}$ and $\rho \in T^\omega$ is an infinite sequence of transitions fireable from $\overline{m}$, then ρ is also fireable from $\overline{n}$. This property is used later on.

We claim that given a reset net $\mathcal{N}$ and an initial marking $\overline{m_0}$, we can build in polynomial time a game G in which $\mathcal{P}_1$ has a winning strategy for objective $\mathsf{BndWMP}_G(0)$ if and only if there exists no infinite execution of the net from $\overline{m_0}$.

We build the game $G = (S_1, S_2, E, k, w)$ with $k = |P| + 3$ as represented in Fig. 10.3 and Fig. 10.4. Unlabeled edges have value zero in all dimensions. For clarity, we define the following $|P|$-dimension integer vectors: $\overline{\mathbf{1}} = (1, \ldots, 1)$ is the unit vector, $\overline{\mathbf{0}} = (0, \ldots, 0)$ is the zero vector, and, for $a, b \in \mathbb{Z}$, $p \in P$, the vector $\overline{\mathbf{a}_{p \to b}}$ represents the vector $(a, \ldots, a, b, a, \ldots, a)$ which has value b in dimension p and a in the other dimensions. The first $|P|$ dimensions encode the tokens present in each place, whereas the last three are used to compel $\mathcal{P}_1$ to act fairly. Our construction ensures that at all times along a valid execution of the net in a gadget, if a place $p \in P$ possess n tokens, then the running sum of weights over the largest open window has value $(-n-1)$ in dimension p.

The states and edges of the game are built as follows.

- Inside a gadget, we have a state fire belonging to $\mathcal{P}_2$, with $|T|$ outgoing edges corresponding to the $|T|$ transitions of the net. Each transition t is encoded as follows:
  - an edge from fire to state test_t belonging to $\mathcal{P}_1$, of value $(\mathbf{I}(t), -1, 0, 0)$,

such that the running sum is updated to accurately encode the consumption of tokens;

- in state test_t, $(|P|+1)$ outgoing edges, giving $\mathcal{P}_1$ the possibility to either branch out of the gadget, going to the state close_p corresponding to the dimension p of his choice, or continuing via an edge of value $(\overline{\mathbf{0}}, -1, 0, 0)$ to the reset_q state, a state of $\mathcal{P}_1$ such that $q = r(t)$ is the unique place reset by transition t;
- a self-loop of value $(\overline{\mathbf{0}_{q\to 1}}, -1, 0, 0)$ on the reset_q state;
- an edge from reset_q to out_t of value $(\overline{\mathbf{0}_{q\to -1}}, -1, 0, 0)$ which purpose is to ensure that in dimension q, there is a new open window of sum -1 after a full reset (i.e., it encodes that the number of tokens in place q is zero);
- an edge from out_t back to fire of value $(-\mathbf{O}(t), -1, 0, 0)$, producing tokens according to the output of transition t.

◦ Branching from the left gadget leads to a state $\mathsf{close}_p^{\text{left}}$ of $\mathcal{P}_1$ with a self-loop of weight $(\overline{\mathbf{1}_{p\to 0}}, 1, 1, -1)$ and an outgoing edge to state $\mathsf{delay}^{\text{left}}$ of $\mathcal{P}_2$.

◦ State $\mathsf{delay}^{\text{left}}$ possess a self-loop of value $(\overline{\mathbf{0}}, 0, 1, 1)$ and an edge going to the right gadget with value $(-\overline{m_0} - \overline{\mathbf{1}}, 0, 0, 0)$.

◦ The right gadget is constructed symmetrically, the only change being that the self-loop on states $\mathsf{close}_p^{\text{right}}$ of $\mathcal{P}_1$ now has value $(\overline{\mathbf{1}_{p\to 0}}, 1, -1, 1)$.

The game starts in the left gadget with an initial edge of value $(-\overline{m_0} - \overline{\mathbf{1}}, 0, 0, 0)$ corresponding to the initial marking of the net.

We claim that (i) if there exists no infinite execution $\rho \in T^\omega$ of the net $\mathcal{N}$, then $\mathcal{P}_1$ has a winning strategy in G for the bounded window objective, and (ii) if there exists such an execution, then $\mathcal{P}_2$ has a winning strategy in G. By determinacy, proving both claims will conclude our proof.

Case (i). Assume that there exists no infinite execution $\rho \in T^\omega$ of the net. Then there exists a bound $b \in \mathbb{N}$ on the length of any valid execution. Hence, $\mathcal{P}_2$ can only simulate the net faithfully for b steps, so after at most $(b+1)$ steps, he needs to use an unfireable transition. That is, the next chosen transition requires more tokens than available in some place $p \in P$. We define a winning strategy $\lambda_1 \in \Lambda_1$ of $\mathcal{P}_1$ in G as follows:

1. In a state test_t, if the last transition t was valid (i.e., all first $|P|$ dimensions have a negative running sum), go to the corresponding reset_q state. Otherwise, there exists a dimension p in which the sum has become non-negative and all windows are closed: exit the gadget and go to the corresponding state close_p.

2. In a state reset_q, cycle until the sum in dimension q takes value 0, then go to state out_t.

3. In a state close_p, take the loop exactly $f(b)$ times before going to state delay, where $f\colon \mathbb{N} \to \mathbb{N}$ is a well-chosen function that we define below (hence $f(b)$ is constant along the play).

We claim that it is possible to define $f(b)$ sufficiently large to ensure that this strategy is winning.

Let $M \in \mathbb{N}$ be the largest number of tokens produced as output of any transition of the net, on any place. We consider the value of the negative sum in any of the first $(|P| + 1)$ dimensions at the moment when $\mathcal{P}_1$ decides to exit the gadget according to the strategy λ_1. Notice that for any dimension $p \in \{1, \ldots, |P|\}$, this sum is bounded by $x = (-\overline{m_0}(p) - 1 - b \cdot M)$. Hence, the number of loops taken on any visit of state reset_q is bounded by x. The sum in dimension $(|P|+1)$ is thus bounded by $(b \cdot (4+x)+1)$, which we define as $f(b)$. The last two dimensions are not modified inside a gadget.

Now clearly, looping in state close_p for $f(b)$ steps is sufficient to close all windows in all dimensions corresponding to places (recall that dimension p is closed by $\mathcal{P}_2$ cheating on place p), as well as in dimension $(|P| + 1)$. However, this loop opens a window in one of the last two dimensions (the last for the left gadget, and the second to last for the right gadget). As the delay state of $\mathcal{P}_2$ has a positive effect in those dimensions, if $\mathcal{P}_2$ decides to delay the play for $f(b)$ steps, all windows will be closed. If he does not delay, the play will proceed to the next gadget, in which $\mathcal{P}_2$ is also forced to cheat before $(b + 1)$ transitions. Hence after looping for $f(b)$ steps in the corresponding close_p state, the open window will close (and another will open in the other dimension which will in turn be closed after the next gadget). By keeping this behavior, $\mathcal{P}_1$ can thus enforce that any open window along the play will close in at most $(4 \cdot f(b) + 4)$ steps. Thus the outcome is winning for the bounded window objective.

Case (ii). Assume that there exists an infinite execution $\rho \in T^\omega$ of the net. We define a winning strategy $\lambda_2 \in \Lambda_2$ of $\mathcal{P}_2$ as follows. The strategy is played in rounds, with the initial round being round 1.

1. Every time a gadget is entered, start playing in state fire according to the infinite execution ρ, that is, choose transitions in order to obtain the same trace.
2. When a state delay is visited during round n, take the self-loop n times then continue to state fire and start round $n+1$.

Notice that this strategy requires infinite memory. We claim that any consistent outcome of the game is winning for $\mathcal{P}_2$, that is, it does not belong to $\mathsf{BndWMP}_G(0)$.

First, $\mathcal{P}_1$ cannot stay forever in a gadget, thanks to dimension $(|P|+1)$: he has to branch at some point otherwise the play is lost. Second, if in state reset_q, $\mathcal{P}_1$ decides to cycle for less than necessary for a full reset, the situation gets better for $\mathcal{P}_2$ by the monotonicity property of the reset net (as $\mathcal{P}_2$ gets to continue with more tokens than expected). Notice that $\mathcal{P}_1$ cannot accumulate positive values in the sum, as the next edge will restart a new window and all accumulation will be forgotten with regard to the objective. Third, if $\mathcal{P}_1$ branches and exits the gadget to go to some state close_p, then all dimensions corresponding to places, including dimension p, have a running open window (dimension p has a strictly negative value since $\mathcal{P}_2$ does not cheat). Hence, no matter how long $\mathcal{P}_1$ chooses the self-loop, the window in dimension p will stay open (and $\mathcal{P}_1$ cannot stay here forever because of the last two dimensions). Fourth, when the play reaches a state delay with an open window in dimension $p \in \{1, \dots, |P|\}$, the strategy λ_2 prescribes that $\mathcal{P}_2$ will loop for longer and longer periods of time, thus enforcing open windows of constantly growing length. As a consequence, any consistent outcome is such that the bounded window objective is not satisfied, which proves our point and further concludes our proof. □

Notice that Theorem 10.10 implies that $\mathcal{P}_1$ may need to use a non-primitive recursive window size to win a multi-dimension bounded window mean-payoff game, whereas a pseudo-polynomial bound exists in the one-player case (see Corollary 9.10). The decidability of the bounded window mean-payoff problem remains open.

10.3 On Direct Objectives

Similarly to what we did in Sect. 9.3 for one-dimension games, we here argue that identical complexity results are obtained for the *direct* variants (Table 10.1), by slight modifications of the presented proofs.

	k-dimension		
	complexity	$\mathcal{P}_1$ mem.	$\mathcal{P}_2$ mem.
direct fixed polynomial window	PSPACE-h. EXP-easy	exponential	
direct fixed arbitrary window	EXP-c.		
direct bounded window problem	NPR-h.	-	-

Table 10.1: Complexities and memory requirements for the direct objectives in multi-dimension games. Results are identical to the prefix-independent case (h. for hard, c. for complete, EXP for EXPTIME and NPR for non-primitive recursive).

Direct fixed window problem. The following results extend to the direct case.

- *EXPTIME algorithm.* Lemma 10.2 presents a reduction from fixed window games to exponentially larger co-Büchi games. It is easy to obtain a similar reduction from direct fixed window games by considering a safety objective for $\mathcal{P}_1$ (i.e., reachability for the set of bad states for $\mathcal{P}_2$). This also implies an exponential-time algorithm.
- *EXPTIME-hardness of the arbitrary window problem for weights* $\{-1, 0, 1\}$ *and arbitrary dimensions.* The reduction of the membership problem for polynomial space alternating Turing machines immediately yields the result for the direct objective. Indeed, the strategies proposed in the proof stay winning for this objective. Note that actually the strategy of $\mathcal{P}_2$ may be simpler, as he may cycle forever on s_{restart} after branching to punish an unfaithful symbol disclosure by keeping a window indefinitely open.

- *EXPTIME-hardness of the arbitrary window problem for two dimensions and arbitrary weights.* The reduction from countdown games established in Lemma 10.6 extends straightforwardly to direct objectives, and $\mathcal{P}_2$ can use a simpler winning strategy consisting in looping forever in its zero cycle.
- *PSPACE-hardness of the polynomial window problem.* The reduction of generalized reachablity games also holds without modification for the direct fixed polynomial window objective.
- *Exponential memory bounds.* Exponential upper bounds follow from the modified Lemma 10.2, using safety games. Lower bounds witnessed by Lemma 10.8 are also verified in the presented game as well as from the reduction of generalized reachablity games.

Direct bounded window problem. Non-primitive-recursive-hardness (Theorem 10.10) extends to the direct objective with a simpler construction. Indeed, it is sufficient to consider the game using only the first $(|P|+1)$ dimensions, and consisting of only one gadget, with the branching out of the gadget now going to an absorbing state with a self-loop of weight $\overline{\mathbf{1}_{p\to 0}}$ such that when $\mathcal{P}_1$ decides to branch, all windows get closed eventually, except in the dimension p of his choice, for which the window is only closed if $\mathcal{P}_2$ cheats and stays open forever otherwise.

Part IV

Beyond Worst-Case Synthesis

CHAPTER 11

The Beyond Worst-Case Framework

Introduction ◇ Beyond Worst-Case Synthesis Problem

We extend the traditional quantitative synthesis framework by going *beyond the worst-case*.

On the one hand, classical analysis of two-player games involves an adversary (modeling the environment of the system) which is *purely antagonistic* and asks for strict worst-case guarantees. On the other hand, MDPs represent situations where the system is faced to a *purely stochastic* environment: the aim is then to optimize the expected payoff, with no guarantee on individual outcomes.

We introduce the *beyond worst-case (BWC) synthesis problem*, which is to construct strategies that guarantee some quantitative requirement in the *worst-case* while providing an higher *expected value* against a particular stochastic model of the environment given as input.

This innovative setting was introduced in joint work with Bruyère, Filiot and Raskin [BFRR13, BFRR14a, BFRR14b].

11.1 Introduction

Classical models. As discussed in Part I of the thesis, two-player zero-sum quantitative games (Sect. 2.1.2) and Markov decision processes (Sect. 2.1.4) are two popular formalisms for modeling decision making in adversarial and uncertain environments respectively.

In the former, two players compete with opposite goals, and we want strategies for $\mathcal{P}_1$ (the system) that ensure a given *minimal performance against all possible strategies* of $\mathcal{P}_2$ (the environment). In the latter, the system plays against a stochastic model of its environment, and we want strategies that ensure a *good expected overall performance*.

Those two models are well studied and simple optimal pure memoryless strategies exist for classical objectives such as mean-payoff or shortest path (Sect. 2.3.2). But both models have clear weaknesses: strategies that are good for the worst-case may exhibit suboptimal behaviors in probable situations while strategies that are good for the expectation may be terrible in some unlikely but possible situations.

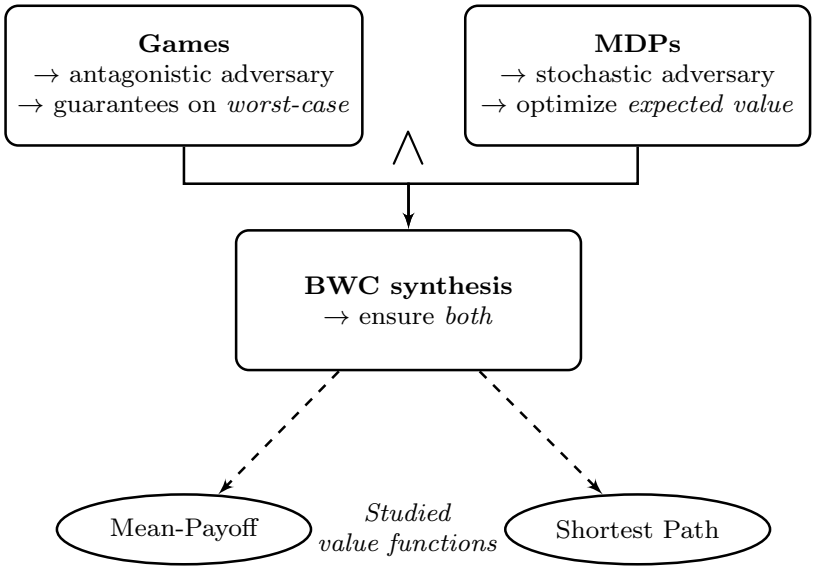


Figure 11.1: The beyond worst-case framework is a crossroad between games and Markov decision processes.

What if we want both? In practice, we would like strategies that both ensure (a) some worst-case threshold no matter how the adversary behaves (i.e., against any arbitrary strategy) and (b) a good expectation against the expected behavior of the adversary (given as a stochastic model).

This is the subject of Part IV, illustrated in Fig. 11.1: we study how to construct finite-memory strategies that ensure both (a) and (b). As motivated in Part II, we are particularly interested in finite memory for $\mathcal{P}_1$ as it can be implemented in practice (as opposed to infinite memory). Note that $\mathcal{P}_2$ is not restricted in his choice of strategies, but we see that simple strategies suffice. Our problem, the *beyond worst-case (BWC) synthesis problem*, makes sense for any quantitative measure. We focus on two classical ones: the *mean-payoff*, and the *shortest path*. Our results are summarized in Sect. 11.1.3.

Illustration. The BWC synthesis problem is relevant to produce system controllers that provide nice expected performance in the everyday situation while ensuring a strict (but relaxed) performance threshold even in the event of very bad (while unlikely) circumstances. We motivate this setting through the following toy example.

Example 11.1. Consider the graph in Fig. 11.2 to illustrate the *shortest path* context. Integer labels are durations in minutes, and fractions are probabilities that model the expected behavior of $\mathcal{P}_2$. This graph can be considered either as a two-player game, if we forget about those probabilities; or as an MDP, if we see states of $\mathcal{P}_2$ as stochastic states with the associated transition probabilities.

Assume $\mathcal{P}_1$ wants a strategy to go from "home" to "work" such that "work" is *guaranteed* to be reached within 60 minutes (to avoid missing an important meeting), and $\mathcal{P}_1$ would also like to minimize the expected time to reach "work".

The strategy that minimizes the expectation is to take the car (expectation is 33 minutes) but it is excluded as there is a possibility to arrive after 60 minutes (in case of heavy traffic). Bicycle is safe but the expectation of this solution is 45 minutes.

We can do better with the following strategy: try to take the train, if the train is delayed three time consecutively, then go back home and take the bicycle. This strategy is safe as it always reaches "work" within 59 minutes and its expectation is $\approx 37,56$ minutes (so better than taking directly the bicycle).

Observe that this simple example already shows that, unlike the situation for

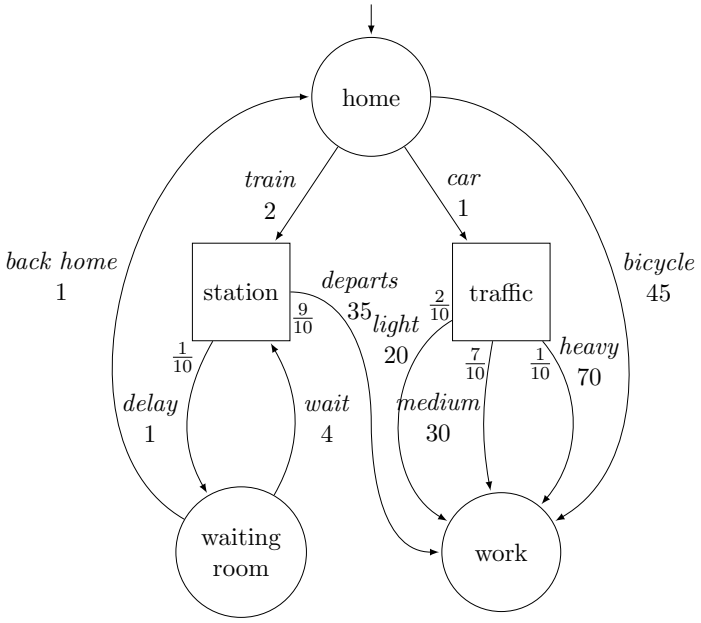


Figure 11.2: Beyond worst-case shortest path - $\mathcal{P}_1$ wants to minimize its expected time to reach "work", but while ensuring it is less than an hour in all cases.

classical games and MDPs, strategies using memory are strictly more powerful than memoryless ones. Our algorithms are able to decide the existence of (and synthesize) such finite-memory strategies. ◁

11.1.1 Notions of Risk-Avoidance

Our BWC problems generalize the corresponding problems for two-player games (worst-case threshold problem) and MDPs (expected value threshold problem).

Our strategies are *strongly risk averse*: they avoid at all cost outcomes below a given threshold (no matter their probability), and inside the set of those *safe* strategies, we maximize expectation. To the best of our knowledge, we are the first to consider such strategies.

Other notions of risk have been studied for MDPs: e.g., in [WL99], the authors want to find policies minimizing the probability (risk) that the total discounted rewards do not exceed a specified value; in [FKR95], the authors want to achieve a specified value of the long-run limiting average reward at a

given probability level (percentile). While those strategies limit risk, they only ensure *low probability* for bad behaviors but not their absence, furthermore, they do not ensure good expectation either.

Another body of related work is the study of strategies in MDPs that achieve a trade-off between expectation and variance over the outcomes (e.g., [BCFK13] for the mean-payoff, [MT11] for the cumulative reward), giving a statistical measure of the stability of the performance. In our setting, we strengthen this requirement by asking for *strict guarantees on individual outcomes*, while maintaining an appropriate expected payoff.

11.1.2 Assumptions and Additional Notations

As usual, we take some time to discuss assumptions taken in Part IV and introduce needed additional notations.

General strategies. The most general class of strategies that we consider is the one of possibly infinite-memory and randomized strategies, denoted Λ. We recall that several subclasses of particular interest are defined in Sect. 2.1.3.

One-dimension games. Throughout Part IV, we restrict our study to one-dimension games. Studying the BWC framework for the more general setting of multi-dimension games is a challenging future work.

Infimum mean-payoff. With regard to the mean-payoff setting, we only study the *infimum variant* of the value function defined in Sect. 2.3.2. Since we mostly deal with finite-memory strategies, this is not restrictive.

Nevertheless, in Sect. 12.7, we show that there is an interesting gap when infinite-memory strategies are considered in the BWC mean-payoff setting. It yields several interesting open questions, and considering the supremum variant of the objective is one of them.

Explicit reference to underlying graphs. Throughout Part IV, we often have to switch from games to MDPs and from MDPs to Markov chains, by fixing strategies for the players, as discussed in Sect. 2.1.6. We then need to be able to refer easily to the corresponding underlying graphs, which may be expanded by the product with SOMMs, as depicted in Fig. 2.5. Therefore, we use the notations of games, MDPs and MCs where those graphs are explicit. For example, $G = (\mathcal{G}, S_1, S_2)$ is a game, with $\mathcal{G} = (S, E, w)$ its underlying graph.

MDPs with edges of probability zero. Recall the definition of MDP as $P = (\mathcal{G}, S_1, S_\Delta, \Delta)$, given in Sect. 2.1.4. In contrast to some other classical definitions of MDPs in the literature, we explicitly allow that, for some states $s \in S_\Delta$, $\mathsf{Supp}(\Delta(s)) \subsetneq \mathsf{Succ}(s)$: some edges of the graph $\mathcal{G}$ are assigned probability zero by the transition function.

This is important as far as modeling is concerned, as in our context, transition functions will be defined according to a stochastic model for the environment of a system, and we cannot reasonably assume that such a model always involves all the possible actions of the environment. Consequently, given the MDP P, we define the subset of edges $E_\Delta = \{(s_1, s_2) \in E \mid s_1 \in S_\Delta \Rightarrow s_2 \in \mathsf{Supp}(\Delta(s_1))\}$, representing all edges that either start in a state of $\mathcal{P}_1$, or are chosen with non-zero probability by the transition function Δ.

End-components. We define end-components of an MDP as subgraphs in which $\mathcal{P}_1$ can ensure to stay despite stochastic states [dA97, BK08].

Definition 11.2. Let $P = (\mathcal{G}, S_1, S_\Delta, \Delta)$ be an MDP, with $\mathcal{G} = (S, E, w)$ its underlying graph. An *end-component* (EC) in P is a set $U \subseteq S$ such that (i) the subgraph $(U, E_\Delta \cap (U \times U))$ is strongly connected, with E_Δ defined as before, i.e., stochastic edges with probability zero are treated as non-existent; and (ii) for all $s \in U \cap S_\Delta$, $\mathsf{Supp}(\Delta(s)) \subseteq U$, i.e., in stochastic states, all outgoing edges either stay in U or belong to $E \setminus E_\Delta$ (that is, the probability of leaving U from a state $s \in S_\Delta$ is zero). The set of all ECs of P is denoted $\mathcal{E} \subseteq 2^S$.

11.1.3 Overview of Results

We study the BWC synthesis problem for two important quantitative settings: the mean-payoff and the shortest path. In both cases, we show how to decide the existence of finite-memory strategies satisfying the problem and how to synthesize one if one exists. We establish algorithms and we study complexity bounds and memory requirements. Our main results are the following.

First, for the mean-payoff, we provide an NP $\cap$ coNP algorithm (Thm. 12.1), which would be in P if mean-payoff games were proved to be in P, a long-standing open problem. Hence, we enrich the modeling and reasoning power over strategies without negative impact on the complexity class of the decision problem. Pseudo-polynomial memory may be necessary and always suffices (Thm. 12.35).

While some memory is necessary, we show that elegantly implementable strategies suffice, constructed using clever alternation between pure memoryless strategies based on intuitive counters. Finally, we observe that infinite-memory strategies are strictly more powerful that finite-memory strategies (Sect. 12.7).

	worst-case	expected value	**BWC**
complexity	NP ∩ coNP	P	**NP ∩ coNP** (Thm. 12.1)
memory	pure memoryless		**pure pseudo-poly.** (Thm. 12.35)

Table 11.1: Overview of decision problem complexities and memory requirements for winning strategies of the first player in games (worst-case), MDPs (expected value) and the BWC setting (combination), for the *mean-payoff*.

Second, for the shortest path, we provide a pseudo-polynomial-time algorithm (Thm. 13.3). We show that the associated decision problem is inherently harder than the worst-case and expected value threshold problems taken separately, as it is NP-hard (Thm. 13.6). Pseudo-polynomial memory may be necessary and always suffices (Thm. 13.4). In the case of the shortest path problem, infinite-memory strategies grant no additional power in comparison with finite-memory strategies (Rem. 13.5).

	worst-case	expected value	**BWC**
complexity	P		**pseudo-poly./NP-hard** (Thm. 13.3/Thm. 13.6)
memory	pure memoryless		**pure pseudo-poly.** (Thm. 13.4)

Table 11.2: Overview of decision problem complexities and memory requirements for winning strategies of the first player in games (worst-case), MDPs (expected value) and the BWC setting (combination), for the *shortest path*.

11.2 Beyond Worst-Case Synthesis Problem

We here define the *beyond worst-case synthesis problem*. Our goal is to study the synthesis of finite-memory strategies that, *simultaneously*, ensure a value greater than some threshold μ in the worst-case situation (i.e., against any strategy of the adversary), and ensure an expected value greater than some threshold ν against a given finite-memory stochastic model of the adversary (e.g., representing commonly observed behavior of the environment).

Definition 11.3. Given a game $G = (\mathcal{G}, S_1, S_2)$, with $\mathcal{G} = (S, E, w)$ its underlying graph, an initial state $s_{\mathsf{init}} \in S$, a finite-memory stochastic model $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^F$ of the adversary, represented by a stochastic output Moore machine, a measurable value function $f \colon \mathsf{Plays}(\mathcal{G}) \to \mathbb{R} \cup \{-\infty, \infty\}$, and two rational thresholds $\mu, \nu \in \mathbb{Q}$, the *beyond worst-case (BWC) problem* asks to decide if $\mathcal{P}_1$ has a finite-memory strategy $\lambda_1 \in \Lambda_1^F$ such that

$$\begin{cases} \forall\, \lambda_2 \in \Lambda_2,\ \forall\, \pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2),\ f(\pi) > \mu & (11.1) \\ \mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1, \lambda_2^{\mathsf{stoch}}]}(f) > \nu & (11.2) \end{cases}$$

and the BWC synthesis problem asks to synthesize such a strategy if one exists.

We take the convention to ask for values strictly greater than the thresholds in order to ease the formulation of our results in the following. Indeed, we will show that for some thresholds, it is possible to synthesize strategies that ensure ε-close values, for any $\varepsilon > 0$, while it is not feasible to achieve the exact threshold (Sect. 12.7). By using the strict inequality in this definition, we avoid tedious manipulation of such ε in our proofs.

Notice that we can assume $\nu > \mu$, otherwise the problem reduces to the classical worst-case analysis as follows. Assume $\mu \geq \nu$ and $\lambda_1^{pm} \in \Lambda_1^{PM}$ satisfies the worst-case threshold (recall memory is not necessary for the worst-case requirement alone). Consider the MC $G[\lambda_1^{pm}, \lambda_2^{\mathsf{stoch}}]$. By eq. (11.1) and Lemma 2.16, we have that for all $\pi \in \mathsf{Outs}_{G[\lambda_1^{pm}, \lambda_2^{\mathsf{stoch}}]}(s_{\mathsf{init}})$, $f(\pi) > \mu$. Hence, regardless of how the probability is defined in the MC, we have that $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^{pm}, \lambda_2^{\mathsf{stoch}}]}(f) > \mu \geq \nu$ and eq. (11.2) is trivially satisfied.

CHAPTER 12

Beyond Worst-Case Mean-Payoff

In a Nutshell ◇ Preprocessing ◇ End-Components Analysis ◇ Inside Winning ECs ◇ Global Strategy ◇ Complexity and Memory Bounds ◇ Infinite Memory

We present an algorithm, BWC_MP (Alg. 12.1), for deciding the BWC mean-payoff problem. Its cornerstones are highlighted in Sect. 12.1.1 and a running example is presented in Sect. 12.1.2.

Sections 12.2 through 12.5 are devoted to a detailed justification of this algorithm and the proof of its correctness.

In Sect. 12.6.1, we prove that algorithm BWC_MP is in NP ∩ coNP and that it is optimal with regard to the complexity of the worst-case problem. In Sect. 12.6.2, we prove that pseudo-polynomial memory is sufficient and in general necessary for finite-memory strategies satisfying the BWC mean-payoff problem.

Finally, we show in Sect. 12.7 that infinite-memory strategies are strictly more powerful than finite-memory ones for $\mathcal{P}_1$. This is in contrast with the worst-case and the expected value settings, where memoryless strategies suffice.

This chapter is based on several joint publications with Bruyère, Filiot and Raskin [BFRR13, BFRR14a, BFRR14b].

12.1 In a Nutshell

12.1.1 The Approach in a Nutshell

Algorithm BWC_MP is described in Alg. 12.1. We give an intuitive sketch of its functioning in the following.

Inputs and outputs. The algorithm takes as input: a game G^i, a finite-memory stochastic model of the adversary λ_2^i, a worst-case threshold μ^i, an expected value threshold ν^i, and an initial state s_{init}^i. Its output is Yes if and only if there exists a finite-memory strategy of $\mathcal{P}_1$ satisfying the BWC problem (Def. 11.3).

The output as described in Alg. 12.1 is boolean: the algorithm answers whether a satisfying strategy exists or not, but does not explicitely construct it (to avoid tedious formalization within the pseudocode). Nevertheless, we present how to synthesize such a winning strategy in Sect. 12.5. We sketch its operation in the following and we highlight the role of each step of the algorithm in the construction of this winning strategy, as producing a witness winning strategy is a straightforward by-product of the process we apply to decide satisfaction of the BWC problem.

Preprocessing. The first part of the algorithm (lines 1 through 10) is dedicated to the preprocessing of the game G^i and the stochatic model λ_2^i given as inputs in order to apply the second part of the algorithm (lines 11 through 17) on a modified game G and stochastic model $\lambda_2^{\mathsf{stoch}}$, simpler to manipulate. We show in the following that the answer to the BWC problem on the modified game is Yes if and only if it is also Yes on the input game, and we present how a winning strategy of $\mathcal{P}_1$ in G can be transferred to a winning strategy in G^i.

The preprocessing is composed of four main steps. First, we modify the weight function of $\mathcal{G}^i$ in order to consider the equivalent BWC problem with thresholds $(0, \nu)$ instead of (μ^i, ν^i). This classical trick is used to get rid of explicitely considering the worst-case threshold in the following, as it is equal to zero.

Second, observe that any strategy that is winning for the BWC problem must also be winning for the classical worst-case problem. Such a strategy cannot allow visits of any state from which $\mathcal{P}_1$ cannot ensure winning against an antagonistic

Algorithm 12.1 BWC_MP$(G^i, \lambda_2^i, \mu^i, \nu^i, s_{\mathsf{init}}^i)$

Require: $G^i = \left(\mathcal{G}^i, S_1^i, S_2^i\right)$ a game, $\mathcal{G}^i = \left(S^i, E^i, w^i\right)$ its underlying graph, $\lambda_2^i \in \Lambda_2^F(G^i)$ a finite-memory stochastic model of the adversary, $\mathcal{M}(\lambda_2^i) = (\mathsf{Mem}, \mathsf{m}_0, \alpha_\mathsf{u}, \alpha_\mathsf{n})$ its SOMM, $\mu^i = \frac{a}{b}, \nu^i \in \mathbb{Q}$, $\mu^i < \nu^i$, resp. the worst-case and the expected value thresholds, and $s_{\mathsf{init}}^i \in S^i$ the initial state

Ensure: The answer is YES if and only if $\mathcal{P}_1$ has a finite-memory strategy $\lambda_1 \in \Lambda_1^F(G^i)$ satisfying the BWC problem from s_{init}^i, for the thresholds pair (μ^i, ν^i) and the mean-payoff value function

{*Preprocessing*}

if $\mu^i \neq 0$ **then**
  Modify the weight function of $\mathcal{G}^i$ s.t. $\forall\, e \in E^i$, $w_{\mathsf{new}}^i(e) := b \cdot w^i(e) - a$, and consider the new thresholds pair $(0, \nu := b \cdot \nu^i - a)$

Compute

$$S_{WC} := \big\{ s \in S^i \mid \exists\, \lambda_1 \in \Lambda_1(G^i), \forall\, \lambda_2 \in \Lambda_2(G^i), \\ \forall\, \pi \in \mathsf{Outs}_{G^i}(s, \lambda_1, \lambda_2), \underline{\mathsf{MP}}(\pi) > 0 \big\}$$

if $s_{\mathsf{init}}^i \notin S_{WC}$ **then**
  return NO
else
  Let $G^w := G^i \restriction S_{WC}$ be the subgame induced by worst-case winning states
  Build $G := G^w \otimes \mathcal{M}(\lambda_2^i) = (\mathcal{G}, S_1, S_2)$, $\mathcal{G} = (S, E, w)$, $S \subseteq (S_{WC} \times \mathsf{Mem})$, the game obtained by product with the SOMM, and $s_{\mathsf{init}} := (s_{\mathsf{init}}^i, \mathsf{m}_0)$ the corresponding initial state
  Let $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^M(G)$ be the memoryless transcription of λ_2^i on G
  Let $P := G[\lambda_2^{\mathsf{stoch}}] = (\mathcal{G}, S_1, S_\Delta = S_2, \Delta = \lambda_2^{\mathsf{stoch}})$ be the MDP obtained from G and $\lambda_2^{\mathsf{stoch}}$

{*Main algorithm*}

Compute $\mathcal{U}_\mathrm{w}$ the set of maximal winning end-components of P
Build $P' = (\mathcal{G}', S_1, S_\Delta, \Delta)$, where $\mathcal{G}' = (S, E, w')$ and w' is defined as follows:

$$\forall\, e = (s_1, s_2) \in E,\ w'(e) := \begin{cases} w(e) \text{ if } \exists\, U \in \mathcal{U}_\mathrm{w} \text{ s.t. } \{s_1, s_2\} \subseteq U \\ 0 \text{ otherwise} \end{cases}$$

Compute the maximal expected value ν^* from s_{init} in P'
if $\nu^* > \nu$ **then**
  return YES
else
  return NO

adversary because mean-payoff is a prefix-independent objective (hence it is not possible to "win" it over the finite prefix up to such a state). Thus, we reduce our study to G^w, the subgame induced by worst-case winning states in G^i (lines 3 and 7). Obviously, if from the initial state s^i_{init}, $\mathcal{P}_1$ cannot win the worst-case problem, then the answer to the BWC problem is No (lines 4-5).

Third, we build the game G which states are defined by the product of the states of G^w and the memory elements of the stochastic output Moore machine $\mathcal{M}(\lambda_2^i)$ (line 8). Intuitively, we expand the initial game by integrating the memory of the stochastic model of $\mathcal{P}_2$ in the graph. Note that this does not modify the power of the adversary.

Fourth, the finite-memory stochastic model λ_2^i on G^i clearly translates to a memoryless stochastic model $\lambda_2^{\mathsf{stoch}}$ on G (line 9). This will help us obtain elegant proofs for the second part of the algorithm.

Analysis of end-components. The second part (lines 11-17) hence operates on a game G such that from all states, $\mathcal{P}_1$ has a strategy to achieve a strictly positive mean-payoff value (recall $\mu = 0$). We consider the MDP $P = G[\lambda_2^{\mathsf{stoch}}]$ and notice that the underlying graphs of G and P are the same thanks to $\lambda_2^{\mathsf{stoch}}$ being memoryless. The following steps rely on the analysis of *end-components* in the MDP, i.e., strongly connected subgraphs in which $\mathcal{P}_1$ can ensure to stay when playing against the stochastic adversary (Def. 11.2).

The motivation to the analysis of ECs is the following. It is well-known that under any arbitrary strategy $\lambda_1 \in \Lambda_1$ of $\mathcal{P}_1$ in P, the probability that states visited infinitely often along an outcome constitute an EC is one [CY95, dA97]. Recall that the mean-payoff is prefix-independent, therefore the value of any outcome only depends on those states that are seen infinitely often. Hence, the expected mean-payoff in $P[\lambda_1]$ depends *uniquely* on the value obtained in the ECs. Inside an EC, we can compute the maximal expected value that can be achieved by $\mathcal{P}_1$, and this value is the same in all states of the EC [FV97].

Consequently, in order to satisfy the expected value requirement (eq. (11.2)), an acceptable strategy for the BWC problem has to favor reaching ECs with a sufficient expectation, but under the constraint that it should also ensure satisfaction of the worst-case requirement (eq. (11.1)). As we show in the following, this constraint implies that some ECs with high expected values may still need to be avoided because they do not permit to guarantee the worst-case requirement.

This is the cornerstone of the classification of ECs that follows.

Classification of end-components. Recall that $\mathcal{E} \subseteq 2^S$ denotes the set of all ECs in P. Notice that by definition, only edges in E_Δ, as defined in Sect. 11.1.2, are involved to determine which sets of states form an EC in P. As such, for any EC $U \in \mathcal{E}$, there may exist edges from $E \setminus E_\Delta$ starting in U, such that $\mathcal{P}_2$ can force leaving U when using an arbitrary strategy. Still these edges will never be used by the stochastic model $\lambda_2^{\mathsf{stoch}}$. This remark will be important to the definition of strategies of $\mathcal{P}_1$ that guarantee the worst-case requirement, as $\mathcal{P}_1$ needs to be able to react to the hypothetic use of such an edge. We will see that it is also the case *inside* an EC.

Now, we want to consider the ECs in which $\mathcal{P}_1$ can ensure that the worst-case requirement will be fulfilled (i.e., without having to leave the EC): we call them *winning* ECs. Indeed, the others will need to be eventually avoided, hence will have zero impact on the expectation of a finite-memory strategy satisfying the BWC problem. So we call the latter *losing* ECs. The subtlety of this classication is that it involves considering the ECs both in the MDP P, and in the game G.

Formally, let $U \in \mathcal{E}$ be an EC. It is *winning* if, in the subgame $G \restriction U$, from all states, $\mathcal{P}_1$ has a strategy to ensure a strictly positive mean-payoff against any strategy of $\mathcal{P}_2$ *that only chooses edges which are assigned non-zero probability by* $\lambda_2^{\mathsf{stoch}}$, or equivalently, edges in E_Δ. We denote $\mathcal{W} \subseteq \mathcal{E}$ the set of such ECs. Non-winning ECs are *losing*: in those, whatever the strategy of $\mathcal{P}_1$ played against the stochastic model $\lambda_2^{\mathsf{stoch}}$ (or any strategy with the same support), there exists at least one outcome for which the mean-payoff is not strictly positive (even if its probability is zero, its mere existence is not acceptable for the worst-case requirement).

Maximal winning end-components. Based on these definitions, observe that line 11 of algorithm BWC_MP does not actually compute the set $\mathcal{W}$ containing all winning ECs, but rather the set $\mathcal{U}_{\mathrm{w}} \subseteq \mathcal{W}$, defined as $\mathcal{U}_{\mathrm{w}} = \{U \in \mathcal{W} \mid \forall U' \in \mathcal{W}, U \subseteq U' \Rightarrow U = U'\}$, i.e., the set of *maximal* winning ECs.

The intuition on *why we can* restrict our study to this subset is as follows. If an EC $U_1 \in \mathcal{W}$ is included in another EC $U_2 \in \mathcal{W}$, i.e., $U_1 \subseteq U_2$, we have that the maximal expected value achievable in U_2 is at least equal to the one achievable in U_1. Indeed, $\mathcal{P}_1$ can reach U_1 with probability one (by virtue of U_2 being an EC and $U_1 \subseteq U_2$) and stay in it forever with probability one (by virtue

of U_1 being an EC): hence the expectation of such a strategy would be equal to what can be obtained in U_1 thanks to the prefix-independence of the mean-payoff. This property implies that it is sufficient to consider maximal winning ECs in our computations.

As for *why we do it*, observe that the complexity gain is critical. The number of winning ECs can be as large as $|\mathcal{W}| \leq |\mathcal{E}| \leq 2^{|S|}$, that is, exponential in the size of the input. Yet, the number of maximal winning ECs is bounded by $|\mathcal{U}_{\mathrm{w}}| \leq |S|$ as they are disjoint by definition. Indeed, for any two winning ECs with a non-empty intersection, their union also constitutes an EC, and is still winning because $\mathcal{P}_1$ can essentially stick to the EC of his choice.

The computation of the set $\mathcal{U}_{\mathrm{w}}$ is executed by a recursive subalgorithm which is in NP $\cap$ coNP. Roughly sketched, this algorithm computes the maximal end-component decomposition of an MDP (in polynomial time [CH12]), then checks for each EC U in the decomposition (their number is polynomial) if U is winning or not, which requires a call to an NP $\cap$ coNP oracle solving the worst-case threshold problem on the corresponding subgame. If U is losing, it may still be the case that a sub-EC $U' \subsetneq U$ is winning. Therefore we recurse on the MDP reduced to U, where states from which $\mathcal{P}_2$ can win in U have been removed (they are a no-go for $\mathcal{P}_1$). Hence the stack of calls is also at most polynomial.

Ensure reaching winning end-components. As discussed, under any arbitrary strategy of $\mathcal{P}_1$, states visited infinitely often form an EC with probability one. Now, if we take a *finite-memory* strategy that *satisfies* the BWC problem (Def. 11.3), we can precise this result and state that they form a *winning* EC with probability one. Equivalently, we have that the probability that an outcome π is such that $\mathsf{Inf}(\pi) = U$ for some $U \in \mathcal{E} \setminus \mathcal{W}$ is zero. The equality is crucial. It may be the case, with non-zero probability, that $\mathsf{Inf}(\pi) = U' \subsetneq U$ for some $U' \in \mathcal{W}$ and $U \in \mathcal{E} \setminus \mathcal{W}$ (hence the recursive algorithm to compute $\mathcal{U}_{\mathrm{w}}$). It is clear that $\mathcal{P}_1$ should not visit all the states of a losing EC forever, as then he would not be able to guarantee the worst-case threshold inside the corresponding subgame.[1]

We denote $S_{\mathsf{neg}} = S \setminus \bigcup_{U \in \mathcal{U}_{\mathrm{w}}} U$ the set of states that, with probability one, are only seen a finite number of times when a BWC satisfying strategy is played,

[1] We show in Sect. 12.7 that with infinite memory, there may still be some incentive to stay in a losing EC.

and call them *negligible* states.

Our ultimate goal here is to build a modified MDP P', sharing the same underlying graph and ECs as P, such that a classical optimal strategy for the expected value problem on P' will naturally avoid losing ECs and prescribe which winning ECs are the most interesting to reach for a BWC strategy on the initial game G and MDP P. Observe that the expected value obtained in P by any BWC satisfying strategy of $\mathcal{P}_1$ only depends on the weights of edges involved in winning ECs, or equivalently, in maximal winning ECs (as the set of outcomes that are not trapped in them has measure zero). Consequently, we build P' by modifying the weight function of P (line 12). Basically, we keep the weights unchanged in edges that belong to some $U \in \mathcal{U}_{\mathrm{w}}$, and we put them to zero everywhere else, i.e., on any edge involving a negligible state. Weight zero is taken because it is lower than the expectation granted by winning ECs, which is strictly greater than zero by definition.

Reach the highest valued winning end-components. We compute the maximal expected mean-payoff ν^* that can be achieved by $\mathcal{P}_1$ in the MDP P', from the corresponding initial state (line 13). This computation takes polynomial time and memoryless strategies suffice to achieve the maximal value (Sect. 2.3.2).

As discussed before, such a strategy reaches an EC of P' with probability one. Basically, we build a strategy that favors reaching ECs with high associated expectations in P'.

We argue that the ECs reached with probability one by this strategy are necessarily winning ECs. Clearly, if a winning EC is reachable instead of a losing one, it will be favored because of the weights definition in P' (expectation is strictly higher in winning ECs). Thus it remains to check if the set of winning ECs is reachable with probability one from any state in S. That is the case because of the preprocessing. Indeed, we know that all states are winning for the worst-case requirement. Clearly, from any state in $A = S \setminus \bigcup_{U \in \mathcal{E}} U$, $\mathcal{P}_1$ cannot ensure to stay in A (otherwise it would form an EC) and thus must be able to win the worst-case requirement from reached ECs. Now for any state in $B = \bigcup_{U \in \mathcal{E}} U \setminus \bigcup_{U \in \mathcal{U}_{\mathrm{w}}} U$, i.e., states in losing ECs and not in any sub-EC winning, $\mathcal{P}_1$ cannot win the worst-case by staying in B, by definition of losing EC. Since we know $\mathcal{P}_1$ can ensure the worst-case by hypothesis, it is clear that he must be able to reach $C = \bigcup_{U \in \mathcal{U}_{\mathrm{w}}} U$ from any state in B, as claimed.

Inside winning end-components. Based on that, winning ECs are reached with probability one. Let us first consider what we can say about such ECs if we assume that $E_\Delta = E$, i.e., if the stochastic model maps all possible edges to non-zero probabilities. We establish a finite-memory *combined strategy* of $\mathcal{P}_1$ that ensures (i) worst-case satisfaction while yielding (ii) an expected value ε-close to the maximal expectation inside the component.

For two well-chosen parameters $K, L \in \mathbb{N}$, it is informally defined as follows: in phase *(a)*, play a memoryless expected value optimal strategy for K steps and memorize $\mathsf{Sum} \in \mathbb{Z}$, the sum of weights along these steps; in phase *(b)*, if $\mathsf{Sum} > 0$, go to *(a)*, otherwise play a memoryless worst-case optimal strategy for L steps, then go to *(a)*. In phases *(a)*, $\mathcal{P}_1$ tries to increase its expectation and approach its optimal one, while in phase *(b)*, he compensates, if needed, losses that occured in phase *(a)*.

The two memoryless strategies exist on the subgame induced by the EC: by definition of ECs, based on E_Δ, the stochastic model of $\mathcal{P}_2$ will never be able to force leaving the EC against the combined strategy.

A key result of our paper is the existence of values for K and L such that (i) and (ii) are verified. We see plays as sequences of periods, each starting with phase *(a)*. First, for any K, it is possible to define $L(K)$ such that any period composed of phases *(a)* + *(b)* ensures a mean-payoff at least $1/(K + L) > 0$. Periods containing only phase *(a)* trivially induce a mean-payoff at least $1/K$ as they are not followed by phase *(b)*. Both rely on the weights being integers. As the length of any period is bounded by $(K + L)$, the inequality remains strict for the mean-payoff of any play, granting (i). Now, consider parameter K. Clearly, when $K \to \infty$, the expectation over a phase *(a)* tends to the optimal one. Nevertheless, phases *(b)* also contribute to the overall expectation of the combined strategy, and (in general) lower it so that it is strictly less than the optimal for any $K, L \in \mathbb{N}$. Hence to prove (ii), we not only need that the probability of playing phase *(b)* decreases when K increases, but also that it decreases faster than the increase of L, needed to ensure (i), so that overall, the contribution of phases *(b)* tends to zero when $K \to \infty$. This is indeed the case and is proved using results bounding the probability of observing a mean-payoff significantly (more than some ε) different than the optimal expectation along a phase *(a)* of length $K \in \mathbb{N}$: this probability decreases exponentially

when K increases [GO02, Tra09] (related to the notions of Chernoff bounds and Hoeffding's inequality in MCs), while L only needs to be polynomial in K.

Now, consider what happens if $E_\Delta \subsetneq E$. Then, if $\mathcal{P}_2$ uses an arbitrary strategy, he can take edges of probability zero, i.e., in $E \setminus E_\Delta$, either staying in the EC, or leaving it. In both cases, this must be taken into account in order to satisfy eq. (11.1) as it may involve dangerous weights (recall that zero-probability edges are not considered when an EC is classified as winning or not). Fortunately, if this were to occur, $\mathcal{P}_1$ could switch to a worst-case winning memoryless strategy, which exists in all states thanks to the preprocessing, to preserve the worst-case requirement. Regarding the expected value (eq. (11.2)), this has no impact as it occurs with probability zero against $\lambda_2^{\mathsf{stoch}}$. The strategy to follow in winning ECs hence adds this reaction procedure to the combined strategy: we call it the *witness-and-secure strategy*.

Global strategy synthesis. In summary, (a) losing ECs should be avoided and will be by a strategy that optimizes the expectation on the MDP P'; (b) in winning ECs, $\mathcal{P}_1$ can obtain the expectation of the EC (at some arbitrarily low ε close) *and* ensure the worst-case threshold.

Hence, we finally compare the value ν^* with the expected value threshold ν (line 14): (i) if it is strictly higher, we conclude that there exists a finite-memory strategy satisfying the BWC problem, and (ii) if it is not, we conclude that there does not exist such a strategy.

To prove (i), we establish a finite-memory strategy in G, called *global strategy*, of $\mathcal{P}_1$ that ensures a strictly positive mean-payoff against an antagonistic adversary, and ensures an expected mean-payoff ε-close to ν^* (hence, strictly greater than ν) against the stochastic adversary modeled by $\lambda_2^{\mathsf{stoch}}$ (i.e., in P). The intuition is as follows. We play the memoryless optimal strategy of the MDP P' for a sufficiently long time, defined by a parameter $N \in \mathbb{N}$, in order to be with probability close to one in a winning EC (the convergence is exponential by results on absorption times in MCs [GS97]). Then, if we are inside a winning EC, we switch to the witness-and-secure strategy which, as sketched in the previous paragraph, ensures the worst-case and the expectation thresholds. If we are not yet in a winning EC, then we switch to a worst-case winning strategy in G, which always exists by hypothesis. Thus the mean-payoff of plays that do not reach winning ECs is strictly positive. Since in winning ECs we are ε-close

to the maximal expected value of the EC, we can conclude that it is possible to play the optimal expectation strategy of MDP P' for sufficiently long to obtain an overall expected value which is arbitrarily close to ν^*, and still guarantee the worst-case threshold in all outcomes.

To prove (ii), it suffices to understand that only ECs have an impact on the expectation, and that losing ECs cannot be used forever without endangering the worst-case requirement. Note that given a winning strategy on G, it is possible to build a corresponding winning strategy on G^i by reintegrating the memory elements of the SOMM in the memory of the winning strategy of $\mathcal{P}_1$.

Complexity bounds. The input size of the algorithm depends on the size of the game, the size of the SOMM for the stochastic model, and the encodings of weights and thresholds. We can prove that all computing steps require (deterministic) polynomial time except for calls to an algorithm solving the worst-case threshold problem, which is in NP∩coNP and not known to be in P (Sect. 2.3.2). Hence, the overall complexity of the algorithm is in NP∩coNP and may collapse to P if the worst-case problem were to be proved in P.

We also establish that the BWC problem is at least as difficult as the worst-case problem thanks to a polynomial-time reduction from the latter to the former. Thus, BWC_MP membership to NP ∩ coNP can be seen as optimal regarding our current knowledge of the worst-case threshold problem.

Theorem 12.1. *The beyond worst-case problem for the mean-payoff value function is in NP∩coNP and at least as hard as deciding the winner in mean-payoff games.*

Remark 12.2 (approximation of the optimal value). Given a worst-case threshold $\mu \in \mathbb{Q}$, a natural question is whether we can maximize the expectation of finite-memory strategies that satisfy this threshold. However, there is no best expectation value in general, as increasing the size of the memory may also strictly increase the expectation. Nevertheless, the least upper bound of all the expected value thresholds that can be achieved by finite-memory strategies can be approached up to an ε, for all $\varepsilon > 0$. Formally, assume that the worst-case threshold μ can be satisfied, and let $\nu_\top$ be the least upper bound of the set $\{\nu \in \mathbb{Q} \mid \exists\, \lambda_1 \in \Lambda_1^{PF}$ that satisfies the BWC problem for thresholds $(\mu, \nu)\}$ (it exists since this set is trivially bounded by W and it is non-empty). Without

knowing $\nu_\top$ a priori, it is still possible to approach it ε-closely, for all $\varepsilon > 0$, by a dichotomic search with a polynomial (in $V = \log_2 W$, the length of the encoding of weights) number of steps, initialized to the interval $[\mu, W]$. ◁

12.1.2 Running Example

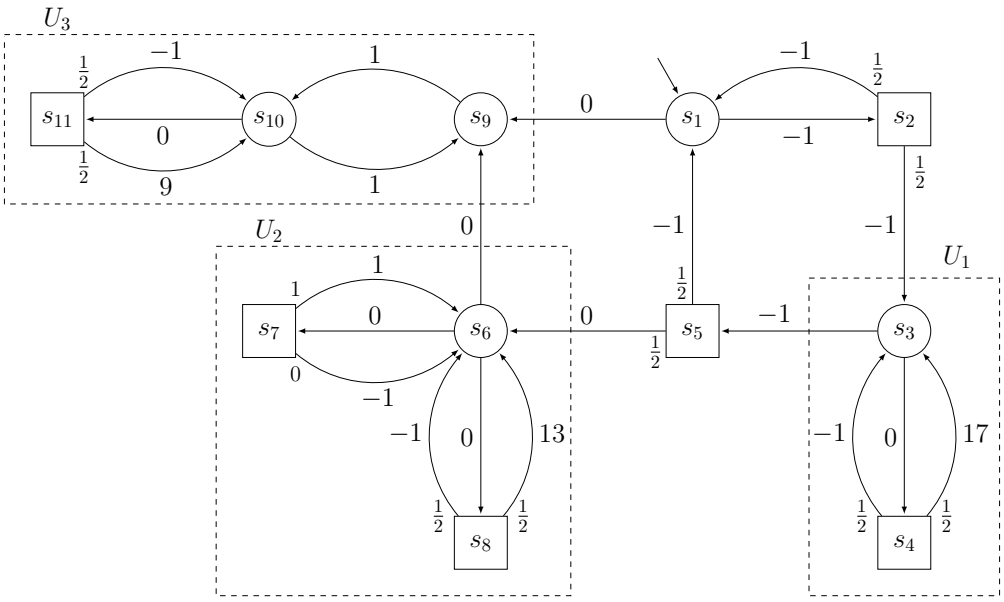


Figure 12.1: Mean-payoff game with maximal winning ECs U_2 and U_3. End-component U_1 is losing.

In order to illustrate several notions and strategies, we will consider the game depicted in Fig. 12.1 throughout this chapter. The stochastic model of $\mathcal{P}_2$ is memoryless and is described by the probabilities written close to the start of outgoing edges. For example, in s_2, the stochastic model chooses edge (s_2, s_1) with probability $1/2$ and edge (s_2, s_3) with probability $1/2$.

Formally, our definition of the set E allows only one edge from any given state s to any state s', hence asks that multiple edges with different values be split by adding dummy states (states with exactly one ingoing edge and one outgoing edge). Note that in order to preserve the same mean-payoff values for paths in the graph, we need to split every edge and copy its weight in both halfs. This restriction is w.l.o.g. and applied for the sake of readability in technical

proofs. Still, our graphical representation oversteps it to maintain compactness.

We consider the BWC problem with the worst-case threshold $\mu = 0$. Observe that this game satisfies the assumptions guaranteed at the end of the preprocessing part of the algorithm. That is, the worst-case threshold is zero, a worst-case winning strategy of $\mathcal{P}_1$ exists in all states (e.g., the memoryless strategy choosing edges (s_1, s_9), (s_3, s_5), (s_6, s_9), (s_9, s_{10}) and (s_{10}, s_9) in their respective starting states), and the stochastic model is memoryless, as explained above.

12.2 Preprocessing - Simplifying Assumptions

We discuss here the preprocessing part of the algorithm (lines 1-10). The goal is to be able to execute the main algorithm (lines 11-17) with the following hypotheses: (a) the worst-case threshold is zero, (b) in all states, $\mathcal{P}_1$ has a strategy to satisfy the worst-case requirement, and (c) the stochastic model of the adversary is memoryless.

This preprocessing is sound and complete: $\mathcal{P}_1$ has a strategy for the BWC problem in the input G^i (for thresholds (μ^i, ν^i) and against stochastic model λ_2^i) if and only if he also has one in the preprocessed game G (for thresholds $(0, \nu)$ and against stochastic model $\lambda_2^{\mathsf{stoch}}$). We break down the proof in three lemmas, one for each hypothesis.

Thresholds. First, Lemma 12.3 states that the worst-case threshold can be taken equal to zero by a slight modification of the weight function (lines 1-2). From now on, we thus assume that $\mu = 0$.

Lemma 12.3. *Let $G = (\mathcal{G}, S_1, S_2)$ be a two-player game, $\mathcal{G} = (S, E, w)$ its underlying graph, $s_{\mathsf{init}} \in S$ the initial state, $\lambda_2^f \in \Lambda_2^F$ a finite-memory stochastic model of $\mathcal{P}_2$, and $(\mu = \frac{a}{b}, \nu) \in \mathbb{Q}^2$ a pair of thresholds, with $a \in \mathbb{Z}$ and $b \in \mathbb{N}_0$. Then $\mathcal{P}_1$ has a satisfying strategy for the BWC mean-payoff problem in G if and only if $\mathcal{P}_1$ has a satisfying strategy when considering the thresholds pair $(0, \nu' = b \cdot \nu - a)$ and the weight function w' such that $\forall\, e \in E$, $w'(e) = b \cdot w(e) - a$.*

Proof. Recall the mean-payoff of a play is defined as the (infimum) limit of the mean weight over prefixes of increasing length. Hence, the affine transformation applied on weights carries over to play values: for all $\pi \in \mathsf{Plays}(\mathcal{G})$, we have that $\underline{\mathsf{MP}}_{w'}(\pi) = b \cdot \underline{\mathsf{MP}}_{w}(\pi) - a$, where the subscript denotes which weight function

we consider. Let $\lambda_1 \in \Lambda_1$ be a strategy of $\mathcal{P}_1$. First, consider the worst-case requirement of eq. (11.1). We claim that

$$\forall\, \lambda_2 \in \Lambda_2,\, \forall\, \pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2),\, \underline{\mathsf{MP}}_w(\pi) > \mu = \frac{a}{b}$$
$$\Updownarrow$$
$$\forall\, \lambda_2 \in \Lambda_2,\, \forall\, \pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2),\, \underline{\mathsf{MP}}_{w'}(\pi) > 0.$$

This is trivial by applying the affine transformation. Second, consider the expected value requirement of eq. (11.2). We claim that

$$\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1,\lambda_2^f]}(\underline{\mathsf{MP}}_w) > \nu \iff \mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1,\lambda_2^f]}(\underline{\mathsf{MP}}_{w'}) > \nu' = b \cdot \nu - a.$$

The expected value operator is well-known to be linear. Thus, by extracting the affine transformation, we obtain that $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1,\lambda_2^f]}(\underline{\mathsf{MP}}_{w'}) = b \cdot \mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1,\lambda_2^f]}(\underline{\mathsf{MP}}_w) - a$, which proves our point and concludes the proof. □

Worst-case winning. Second, we prove in Lemma 12.4 that a necessary condition for a strategy to satisfy the BWC problem is to avoid visiting states that are losing for the worst-case mean-payoff requirement. This justifies lines 3-7 of the algorithm. Observe that the graph of $G^w = G^i \restriction S_{WC}$ contains no deadlock as otherwise it would contradict the fact that $\mathcal{P}_1$ can satisfy the worst-case threshold problem from states in S_{WC} in the game G^i. Also note that $G^w[\lambda_2^i]$ remains a well-defined MDP as there exists no edge from states $s \in S_2^i \cap S_{WC}$ to states $s' \in S \setminus S_{WC}$, otherwise $\mathcal{P}_2$ could win the game from s by reaching s', by prefix-independence of the mean-payoff, and so s would no belong to S_{WC}.

Lemma 12.4. *Let $G = (\mathcal{G}, S_1, S_2)$ be a two-player game, $\mathcal{G} = (S, E, w)$ its underlying graph, $s_{\mathsf{init}} \in S$ the initial state, $\lambda_2^f \in \Lambda_2^F$ a finite-memory stochastic model of $\mathcal{P}_2$, and $\nu \in \mathbb{Q}$ the expected value threshold. Let $S_{WC} \subseteq S$ be the set of winning states for $\mathcal{P}_1$ for the worst-case threshold problem. If $\lambda_1 \in \Lambda_1^F$ is a satisfying strategy for the BWC mean-payoff problem, then*

$$\forall\, \lambda_2 \in \Lambda_2,\, \forall\, \pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2),\, \forall\, s \in S \setminus S_{WC},\, s \notin \pi. \tag{12.1}$$

Proof. Formally, let

$$S_{WC} := \{s \in S \mid \exists\, \lambda_1 \in \Lambda_1, \forall\, \lambda_2 \in \Lambda_2, \forall\, \pi \in \mathsf{Outs}_G(s, \lambda_1, \lambda_2), \underline{\mathsf{MP}}(\pi) > 0\}\,.$$

We claim that a winning strategy of $\mathcal{P}_1$ for the BWC problem must avoid states that are not in S_{WC} and prove it by contradiction. Indeed, let $\lambda_1 \in \Lambda_1^F$ be such a strategy. Assume eq. (12.1) is false: there exist $\lambda_2 \in \Lambda_2$, $\pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2)$ and $s \in S \setminus S_{WC}$ such that $s \in \pi$. By definition of S_{WC} and determinacy of mean-payoff games [EM79, Mar75], it is the case that in state s, $\mathcal{P}_2$ has a winning strategy for the worst-case objective: there exists $\lambda_2' \in \Lambda_2$ such that for all $\lambda_1' \in \Lambda_1$, there exists $\pi' \in \mathsf{Outs}_G(s, \lambda_1', \lambda_2')$ such that $\neg(\underline{\mathsf{MP}}(\pi') > 0)$, i.e., $\underline{\mathsf{MP}}(\pi') \leq 0$. Hence, consider the strategy λ_2'' of $\mathcal{P}_2$ that plays according to λ_2 up to the first visit of s and according to λ_2' afterwards. Clearly, there exists an outcome $\pi'' \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2'')$ such that $\pi'' = \rho \cdot \pi'$ and $\underline{\mathsf{MP}}(\pi') \leq 0$ for some $\rho \in \mathsf{Prefs}(\mathcal{G})$. Since the mean-payoff objective is prefix-independent, we have that $\underline{\mathsf{MP}}(\pi'') \leq 0$. Thus, the strategy λ_1 of $\mathcal{P}_1$ does not satisfy eq. (11.1), which contradicts the hypothesis and concludes the proof. □

Memoryless stochastic model. Finally, we show in Lemma 12.5 that we can study the equivalent BWC problem on the game obtained by product of the original game and the SOMM of the stochastic model, using a memoryless stochastic model instead of the finite-memory one (lines 8-10 of BWC_MP). Having a memoryless stochastic model proves useful in the main steps of the algorithm (lines 11-17) as it guarantees that the game G and the MDP $G[\lambda_2^{\mathsf{stoch}}]$ possess the same underlying graph.

Lemma 12.5. *Let $G = (\mathcal{G}, S_1, S_2)$ be a two-player game, $\mathcal{G} = (S, E, w)$ its underlying graph, $s_{\mathsf{init}} \in S$ the initial state, $\lambda_2^f \in \Lambda_2^F(G)$ a finite-memory stochastic model of $\mathcal{P}_2$, $\mathcal{M}(\lambda_2^f) = (\mathsf{Mem}, \mathsf{m}_0, \alpha_{\mathsf{u}}, \alpha_{\mathsf{n}})$ its SOMM, and $\nu \in \mathbb{Q}$ the expected value threshold. Let $G' = G \otimes \mathcal{M}(\lambda_2^f)$ be the product game and $\lambda_2^m \in \Lambda_2^M(G')$ the memoryless transcription of λ_2^f on G'. The two following statements are equivalent.*

(a) $\mathcal{P}_1$ has a strategy to satisfy the BWC problem on G against the finite-memory stochastic model λ_2^f.

(b) $\mathcal{P}_1$ *has a strategy to satisfy the BWC problem on* G' *against the memoryless stochastic model* λ_2^m.

Proof. We define the product game $G' = G \otimes \mathcal{M}(\lambda_2^f) = (\mathcal{G}', S_1', S_2')$, with $\mathcal{G}' = (S', E', w')$, as follows.

- $S' = S \times \mathsf{Mem}$, $S_1' = S_1 \times \mathsf{Mem}$, $S_2' = S_2 \times \mathsf{Mem}$;
- $\forall\, s_1, s_2 \in S, \forall\, \mathsf{m}_1, \mathsf{m}_2 \in \mathsf{Mem}$, $((s_1, \mathsf{m}_1), (s_2, \mathsf{m}_2)) \in E' \Leftrightarrow (s_1, s_2) \in E \wedge \alpha_{\mathsf{u}}(\mathsf{m}_1, s_1) = \mathsf{m}_2$;
- $\forall\, e = ((s_1, \mathsf{m}_1), (s_2, \mathsf{m}_2)) \in E'$, $w'(e) = w((s_1, s_2))$;
- $s'_{\mathsf{init}} = (s_{\mathsf{init}}, \mathsf{m}_0)$ is the new initial state.

Given the finite-memory stochastic model $\lambda_2^f \in \Lambda_2^F(G)$ of $\mathcal{P}_2$, we transcript it into a memoryless strategy $\lambda_2^m \in \Lambda_2^M(G')$ on the product game such that

$$\forall\, s_1 \in S_2,\ \forall\ ((s_1, \mathsf{m}_1), (s_2, \mathsf{m}_2)) \in E',\ \lambda_2^m((s_1, \mathsf{m}_1))((s_2, \mathsf{m}_2)) = \alpha_{\mathsf{n}}(\mathsf{m}_1, s_1)(s_2).$$

Basically, we have integrated the finite memory of λ_2^f into the states of G' and defined the remaining corresponding memoryless strategy λ_2^m on G'.

We first prove that *(a)* $\Rightarrow$ *(b)*. Assume $\lambda_1 \in \Lambda_1^F(G)$ is a satisfying strategy for the BWC problem on G, i.e., it satisfies eq. (11.1) and (11.2) against the stochastic model λ_2^f. We build a corresponding strategy $\lambda_1' \in \Lambda_1^F(G')$ that is winning against λ_2^m in G' as follows:

$$\begin{gathered}\forall\, \rho' = (s_0, \mathsf{m}_0)(s_1, \mathsf{m}_1)\dots(s_k, \mathsf{m}_k) \in \mathsf{Prefs}_1(G'),\ (s_{k+1}, \mathsf{m}_{k+1}) \in S' \\ \text{such that } ((s_k, \mathsf{m}_k), (s_{k+1}, \mathsf{m}_{k+1})) \in E', \\ \lambda_1'(\rho')\,((s_{k+1}, \mathsf{m}_{k+1})) = \lambda_1(\mathsf{proj}_S(\rho'))(s_{k+1}).\end{gathered}$$

Note that strategy λ_1' is well-defined as λ_1 is. Consider the worst-case requirement of the BWC problem on G', eq. (11.1). Let $\pi' \in \mathsf{Outs}_{G'}(s'_{\mathsf{init}}, \lambda_1')$ be any outcome consistent with the newly defined strategy λ_1'. By definition of λ_1', we have that $\pi = \mathsf{proj}_S(\pi') \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1)$ is an outcome consistent with λ_1 in G. Hence, by hypothesis, we have that $\underline{\mathsf{MP}}(\pi) > 0$. By definition of w', we have that $\underline{\mathsf{MP}}_{w'}(\pi') = \underline{\mathsf{MP}}_w(\pi)$, thus $\underline{\mathsf{MP}}(\pi') > 0$ and the worst-case requirement is satisfied. Now consider the expected value, eq. (11.2). By hypothesis, we have

that $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1,\lambda_2^f]}(\underline{\mathsf{MP}}) > \nu$. We claim that $\mathbb{E}_{s'_{\mathsf{init}}}^{G'[\lambda'_1,\lambda_2^m]}(\underline{\mathsf{MP}}) > \nu$. Indeed, observe that there is a bijection between outcomes in $\mathsf{Outs}_{G[\lambda_1,\lambda_2^f]}(s_{\mathsf{init}})$ and $\mathsf{Outs}_{G'[\lambda'_1,\lambda_2^m]}(s'_{\mathsf{init}})$ using the projection operator because the memory update function α_{u} of the SOMM $\mathcal{M}(\lambda_2^f)$ is deterministic. As the values of plays are preserved by the changes to w' and the probability measures of cylinder sets are also preserved by definition of λ_2^m, the claim is verified.

Second, we show that *(b)* $\Rightarrow$ *(a)*. Assume $\lambda'_1 \in \Lambda_1^F(G')$ is a satisfying strategy for the BWC problem on G', against the stochastic model λ_2^m. We build a corresponding strategy $\lambda_1 \in \Lambda_1^F(G)$ that is winning against λ_2^f in G. Thanks to the update function of $\mathcal{M}(\lambda_2^f)$ being deterministic, given a prefix $\rho = s_0 s_1 \dots s_k \in \mathsf{Prefs}(G)$, there is a unique corresponding prefix $\rho' \in \mathsf{Prefs}(G')$ such that $\rho = \mathsf{proj}_S(\rho')$. Hence we define λ_1 as follows:

$$\forall\, \rho = s_0 s_1 \dots s_k \in \mathsf{Prefs}_1(G),\ s_{k+1} \in S \text{ such that } (s_k, s_{k+1}) \in E,$$
$$\lambda_1(\rho)(s_{k+1}) = \lambda'_1(\rho')((s_{k+1}, \mathsf{m}_{k+1})),$$

with $\rho' = (s_0, \mathsf{m}_0)(s_1, \mathsf{m}_1)\dots(s_k, \mathsf{m}_k)$ the unique prefix in $\mathsf{Prefs}_1(G')$ such that $\rho = \mathsf{proj}_S(\rho')$ and $\mathsf{m}_{k+1} = \alpha_{\mathsf{u}}(\mathsf{m}_k, s_k)$. Strategy λ_1 is well-defined. Consider the worst-case requirement. Let $\pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1)$ be any consistent outcome and π' the unique corresponding play in G'. By definition, $\pi' \in \mathsf{Outs}_{G'}(s'_{\mathsf{init}}, \lambda'_1)$. Hence, by construction, $\underline{\mathsf{MP}}(\pi) = \underline{\mathsf{MP}}(\pi')$, and by hypothesis, $\underline{\mathsf{MP}}(\pi') > 0$, which proves that λ_1 satisfies eq. (11.1) in G. Finally, consider the expected value requirement, eq. (11.2). Again, there is a bijection between $\mathsf{Outs}_{G[\lambda_1,\lambda_2^f]}(s_{\mathsf{init}})$ and $\mathsf{Outs}_{G'[\lambda'_1,\lambda_2^m]}(s'_{\mathsf{init}})$. Since weights and probability measures are preserved, we have that $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1,\lambda_2^f]}(\underline{\mathsf{MP}}) = \mathbb{E}_{s'_{\mathsf{init}}}^{G'[\lambda'_1,\lambda_2^m]}(\underline{\mathsf{MP}}) > \nu$. This concludes our proof that assertions *(a)* and *(b)* are equivalent. □

Note that given a satisfying strategy in the product game, the proof of Lemma 12.5 describes how to obtain a corresponding satisfying strategy in the original game. Hence, strategies obtained through algorithm BWC_MP are preserved alongside the answer to the BWC problem when going back to the original game.

We now study the main steps of algorithm BWC_MP, assuming the simplifying assumptions granted by the preprocessing.

12.3 End-Components Analysis

12.3.1 Classification of End-Components

In the following, we consider a game G where all states are winning for the worst-case requirement (eq. (11.1)), a memoryless stochastic model $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^M$, and $P = G[\lambda_2^{\mathsf{stoch}}]$, the MDP obtained when this stochastic model is followed by $\mathcal{P}_2$. As sketched in Sect. 12.1.1, the crux of the algorithm is the analysis of the end-components of P.

Long-run appearance of end-components. Lemma 12.6 recalls a well-known property of MDPs: under any arbitrary strategy of $\mathcal{P}_1$, the set of states visited infinitely often along a play almost-surely constitutes an EC. Notice the abuse of notation as discussed in Sect. 2.1.6.

Lemma 12.6 ([CY95, dA97]). *Let $P = (\mathcal{G}, S_1, S_\Delta, \Delta)$ be an MDP, with $\mathcal{G} = (S, E, w)$ its underlying graph, $\mathcal{E} \subseteq 2^S$ the set of its end-components, $s_{\mathsf{init}} \in S$ the initial state, and $\lambda_1 \in \Lambda_1(P)$ an arbitrary strategy of $\mathcal{P}_1$. Then, we have that*

$$\mathbb{P}_{s_{\mathsf{init}}}^{P[\lambda_1]}\left(\{\pi \in \mathsf{Outs}_{P[\lambda_1]}(s_{\mathsf{init}}) \mid \mathsf{Inf}(\pi) \in \mathcal{E}\}\right) = 1.$$

Hence the expected value $\mathbb{E}_{s_{\mathsf{init}}}^{P[\lambda_1]}(\underline{\mathsf{MP}})$ depends exclusively on the values of end-components (because the mean-payoff of any play belongs to $[-W, W]$ and thus the value of plays not entering ECs, which set has probability measure zero, cannot be infinite).

Winning end-components. First, we introduce some notations. We respectively denote G_Δ and P_Δ the game and MDP where the underlying graph is limited to the subset of edges which are assigned non-zero probability by $\Delta = \lambda_2^{\mathsf{stoch}}$, i.e., $E_\Delta \subseteq E$. By definition, ECs are computed with regard to E_Δ, hence the ECs of P_Δ are exactly equal to the ECs of P. Moreover, we have that

$$\Lambda_2(G_\Delta) = \left\{\lambda_2 \in \Lambda_2(G) \mid \forall\, \rho \cdot s \in \mathsf{Prefs}_2(G),\, \mathsf{Supp}(\lambda_2(\rho \cdot s)) \subseteq \mathsf{Supp}(\Delta(s))\right\},$$

whereas available choices are unchanged for $\mathcal{P}_1$ in G_Δ as edges in $E \setminus E_\Delta$ all start in states of $\mathcal{P}_2$.

Using these notations, we now define *winning* ECs as the ECs $U \in \mathcal{E}$ where $\mathcal{P}_1$ can ensure satisfaction of the worst-case requirement in the corresponding sub-

game $G_\Delta \restriction U$. Note that we consider G_Δ as we only need to consider strategies of $\mathcal{P}_2$ that share the support of the stochastic model. That is because our goal is to ensure that $\mathcal{P}_1$ can benefit from (the maximal expectation achievable in) these ECs, which is only safe with regard to the worst-case requirement if $\mathcal{P}_1$ can force that all outcomes that may occur when facing the stochastic model yield a strictly positive mean-payoff. Note that reacting to an arbitrary strategy of $\mathcal{P}_2$, i.e., a strategy in $\Lambda_2(G)$, as required in eq. (11.1), will be considered in the following sections: for now we only care about $\lambda_2^{\mathsf{stoch}}$ and satisfaction of the expected value requirement as specified in eq. (11.2).

Definition 12.7. Let $G = (\mathcal{G}, S_1, S_2)$ be a two-player game, $\mathcal{G} = (S, E, w)$ its underlying graph, $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^M$ a memoryless stochastic model of $\mathcal{P}_2$, $P = G[\lambda_2^{\mathsf{stoch}}] = (\mathcal{G}, S_1, S_\Delta = S_2, \Delta = \lambda_2^{\mathsf{stoch}})$ the resulting MDP and G_Δ the game reduced to non-zero probability edges. Let $U \in \mathcal{E}$ be an end-component of P. Then, we have that

- $U \in \mathcal{W}$, the *winning ECs* (WECs), if

$$\begin{aligned} \exists\, \lambda_1 \in \Lambda_1(G_\Delta \restriction U), \forall\, \lambda_2 \in \Lambda_2(G_\Delta \restriction U),\ \forall\, s \in U, \\ \forall\, \pi \in \mathsf{Outs}_{(G_\Delta \restriction U)}(s, \lambda_1, \lambda_2),\ \underline{\mathsf{MP}}(\pi) > 0\ ; \end{aligned} \tag{12.2}$$

- $U \in \mathcal{L}$, the *losing ECs* (LECs), otherwise. By determinacy of mean-payoff games,

$$\begin{aligned} \exists\, \lambda_2 \in \Lambda_2(G_\Delta \restriction U), \forall\, \lambda_1 \in \Lambda_1(G_\Delta \restriction U),\ \exists\, s \in U, \\ \exists\, \pi \in \mathsf{Outs}_{(G_\Delta \restriction U)}(s, \lambda_1, \lambda_2),\ \underline{\mathsf{MP}}(\pi) \leq 0. \end{aligned} \tag{12.3}$$

Note that an EC is winning if $\mathcal{P}_1$ has a worst-case winning strategy from *all* states. This point is important as it may well be the case that winning strategies exist in a strict subset of states of the EC. This does not contradict the definition of ECs as strongly connected subgraphs, as the latter only guarantees that every state can be reached *with probability one*, and not necessarily surely. Hence one cannot call upon the prefix-independence of the mean-payoff to extend the existence of a winning strategy to all states.

Such a situation can be observed on the game of Fig. 12.2, where the EC U_2 is losing (because from s_1, the outcome $(s_1 s_3 s_4)^\omega$ can be forced by $\mathcal{P}_2$, yielding

mean-payoff $-1/3 \leq 0$), while its sub-EC U_3 is winning. From s_1, $\mathcal{P}_1$ can ensure to reach U_3 almost-surely, but not surely, which is critical in this case.

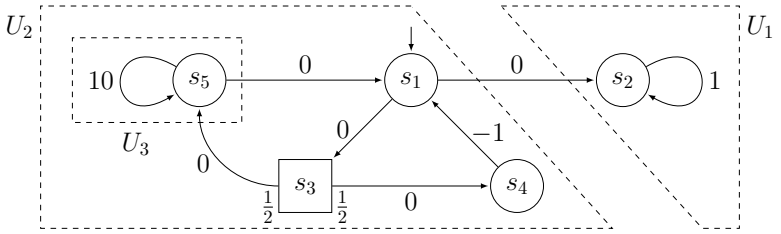


Figure 12.2: End component U_2 is losing. The set of maximal winning ECs is $\mathcal{U}_{\mathrm{w}} = \mathcal{W} = \{U_1, U_3\}$.

Maximality. As discussed in Sect. 12.1.1, we can restrict our analysis to *maximal* winning ECs in the following. This is a consequence of Lemma 12.8.

Lemma 12.8. *Let $U_1, U_2 \in \mathcal{W}$ be two winning ECs in the MDP P such that $U_1 \subsetneq U_2$. Let ν_1^*, ν_2^* denote the respective maximal expected values achievable by $\mathcal{P}_1$ in U_1 and U_2. Then, we have that $\nu_1^* \leq \nu_2^*$.*

Proof. First notice that the maximal expectation achievable in an EC does not depend on the starting state inside the EC. Hence, assume any state $s_2 \in U_2$. Since $U_1 \subsetneq U_2$ and U_2 is an end-component, $\mathcal{P}_1$ can reach a state in U_1 with probability one from s_2. Since the mean-payoff value function only takes finite values, the contribution of plays that do not reach U_1 in the expected value is null. Finally, by prefix-independence of the mean-payoff, we can forget about the finite prefixes outside U_1 and we deduce that $\nu_2^* \geq \nu_1^*$. □

We formally define the set of *maximal winning ECs* as

$$\mathcal{U}_{\mathrm{w}} = \{U \in \mathcal{W} \mid \forall\, U' \in \mathcal{W},\, U \subseteq U' \Rightarrow U = U'\}.$$

This set is not to be confused with the set of winning maximal ECs, $\{U \in \mathcal{W} \mid \forall\, U' \in \mathcal{E},\, U \subseteq U' \Rightarrow U = U'\} \subseteq \mathcal{U}_{\mathrm{w}}$, which holds no particular interest for us. While the total number of winning ECs $|\mathcal{W}| \leq |\mathcal{E}| \leq 2^{|S|}$ can be exponential in the number of states of the game, the number of maximal winning ECs $|\mathcal{U}_{\mathrm{w}}| \leq |S|$ is bounded by this number of states, as all ECs of $\mathcal{U}_{\mathrm{w}}$ are disjoint (because the union of two winning ECs is itself a winning EC). Hence, restriction to the

maximal winning ECs is a cornerstone in the overall NP ∩ coNP complexity that we claim for algorithm BWC_MP.

Illustration. Consider the running example in Fig. 12.1. Note that states s_1, s_2 and s_5 do not belong to any EC: given any strategy of $\mathcal{P}_1$ in P, with probability one, any consistent outcome will only visit those states a finite number of times (Lemma 12.6). The set of *maximal winning ECs* is $\mathcal{U}_{\mathrm{W}} = \{U_2, U_3\}$. Obviously, those ECs are disjoint. The set of winning ECs is much larger, $\mathcal{W} = \mathcal{U}_{\mathrm{W}} \cup \{\{s_9, s_{10}\}, \{s_{10}, s_{11}\}, \{s_6, s_7\}, \{s_6, s_8\}\}$.

End-component U_1 is *losing*. Indeed, in the subgame $G_\Delta \restriction U_1$, the strategy consisting in always picking the -1 edge guarantees an outcome which mean-payoff is negative. Note that this edge is present in E_Δ as it is assigned probability $1/2$ by the stochastic model. Here, we witness why it is important to base our definition of winning ECs on the game G_Δ rather than G. Indeed, in $G \restriction U_2$, it is also possible for $\mathcal{P}_2$ to guarantee a negative mean-payoff by always choosing edges with weight -1. However, to achieve this, $\mathcal{P}_2$ has to pick edges that are in E_Δ: this will never happen against the stochastic model and as such, this can be watched by $\mathcal{P}_1$ to see if $\mathcal{P}_2$ uses an arbitrary antagonistic strategy, and dealt with. If $\mathcal{P}_2$ conforms to E_Δ, i.e., if he plays in G_Δ, he has to pick the edge of weight 1 in s_7 and $\mathcal{P}_1$ has a worst-case winning strategy consisting in always choosing to go in s_7. This EC is thus classified as *winning*. Note that for U_3, in both subgames $G \restriction U_3$ and $G_\Delta \restriction U_3$, $\mathcal{P}_1$ can guarantee a strictly positive mean-payoff by playing $(s_9\, s_{10})^\omega$: even *arbitrary* strategies of $\mathcal{P}_2$ cannot endanger $\mathcal{P}_1$ in this case.

Lastly, consider the game depicted in Fig. 12.2. While U_2 is a strict superset of U_3, the former is losing whereas the latter is winning, as explained above. Hence, the set $\mathcal{U}_{\mathrm{W}}$ is equal to $\{U_1, U_3\}$.

Computation of the maximal winning end-components. Obviously, from a complexity standpoint, to benefit from the polynomial size of $\mathcal{U}_{\mathrm{W}}$, in contrast to the potentially exponential size of $\mathcal{W}$, we need to compute $\mathcal{U}_{\mathrm{W}}$ without first computing all winning ECs of $\mathcal{W}$. We present an algorithm to do so, called MWEC, in Alg. 12.2. Lemma 12.9 establishes that MWEC is correct and complete, and is in NP ∩ coNP, resorting to an NP ∩ coNP oracle to solve mean-payoff games (i.e., decide the answer of the worst-case threshold problem), other than that implementing polynomial operations. It operates under the assumption that all

Algorithm 12.2 MWEC(P)

Require: $P = (\mathcal{G}, S_1, S_\Delta, \Delta)$, with $\mathcal{G} = (S, E, w)$ such that $E_\Delta = E$
Ensure: $\mathcal{M}_\mathrm{W} = \mathcal{U}_\mathrm{W}$, the set of maximal winning ECs of P

1: **if** P is empty **then**
2: **return** $\emptyset$
3: **else**
4: Compute $\mathcal{M}_\mathrm{W} := \{U_1, \dots, U_n\}$ the maximal EC decomposition of P
5: **for all** $i = 1, \dots, n$ **do**
6: Compute $L_i \subseteq U_i$ the set of states from which $\mathcal{P}_2$ has a strategy to enforce $\underline{\mathsf{MP}} \leq 0$ in the subgame $G \restriction U_i$, i.e.,

$$L_i := \{s \in U_i \mid \forall\, \lambda_1 \in \Lambda_1(P \restriction U_i), \exists\, \pi \in \mathsf{Outs}_{P \restriction U_i}(s, \lambda_1), \underline{\mathsf{MP}}(\pi) \leq 0\}$$

7: **if** $L_i \neq \emptyset$ **then**
8: $\mathcal{M}_\mathrm{W} := (\mathcal{M}_\mathrm{W} \setminus \{U_i\}) \cup \mathrm{MWEC}(P \restriction (U_i \setminus L_i))$
9: **return** $\mathcal{M}_\mathrm{W}$

edges are of non-zero probability, i.e., $E_\Delta = E$. This is w.l.o.g. as it suffices to remove those edges as a preprocessing step (they have no impact in the definitions of ECs and winning ECs).

Lemma 12.9. *Let $P = (\mathcal{G}, S_1, S_\Delta, \Delta)$ be an MDP, with $\mathcal{G} = (S, E, w)$ its underlying graph such that $E_\Delta = E$. Then algorithm* MWEC *computes its set of maximal winning ECs $\mathcal{U}_\mathrm{W} = \mathrm{MWEC}(P)$ and is in NP $\cap$ coNP.*

Algorithm MWEC can be sketched as follows. Given a non-empty MDP, it first computes its decomposition into maximal end-components[2] (without distinction between winning and losing ECs). This can be obtained in polynomial time [CH12]. Afterwards, it checks for each of these ECs if it is winning or not, in the sense of Def. 12.7. If the EC is winning, it is now part of the set of claimed maximal winning ECs, denoted $\mathcal{M}_\mathrm{W}$ in the algorithm, and the algorithm will not recurse on this set of states. If the EC is losing, then it may still be the case that a sub-EC is winning, as discussed earlier. Hence, the algorithm eliminates all worst-case losing states and executes recursively on the induced subgame. It

[2]Given an MDP P with a set of ECs $\mathcal{E}$, an EC U is said to be *maximal* in P if for all $U' \in \mathcal{E}$, $U \subseteq U' \Rightarrow U = U'$. This is not to be confused with the definition of maximal winning ECs, given in this section. In particular, maximal ECs need not be winning in general, whereas maximal winning ECs need not be maximal ECs in the sense we just defined (they only need to be maximal with regard to other *winning* ECs).

stops recursing on sub-MDPs whenever one is declared winning or empty. Since it suppresses at least one state in each call, the algorithm is ensured to stop. Moreover, the number of calls is polynomial in the size of the MDP. Deciding if an EC is winning or losing requires calling an NP ∩ coNP oracle solving the worst-case threshold problem (see Sect. 2.3.2).

The remainder of this section is dedicated to the proof of the correctness and completeness of the algorithm, as well as an illustration of its operation. We first state several remarks on its functioning that will be of importance in the proof.

Remark 12.10. In line 6, we have that $P \restriction U_i$ is a well-defined MDP, since U_i is an EC. Similarly, in line 8, $P \restriction (U_i \setminus L_i)$ is also a well-defined MDP. Indeed, from all states $s \in (U_i \setminus L_i) \cap S_1$, there exists an edge from s that goes to a state of $U_i \setminus L_i$, otherwise s would be a losing state (and so would be in L_i). Moreover, for all states $s \in (U_i \setminus L_i) \cap S_\Delta$, there is no edge from s that goes in L_i (otherwise s would be in L_i) nor in $S \setminus U_i$ (otherwise U would not be an EC), and therefore the probability distribution $\Delta(s)$ is still well-defined on $P \restriction (U_i \setminus L_i)$. Additionally, this sub-MDP still verifies that there is no edge with probability zero. ◁

Remark 12.11. Let U be an EC of P, let L be its set of losing states (as computed by line 6), and let $V \subseteq U \setminus L$. Then V is a winning EC of $P \restriction (U \setminus L)$ if and only if V is a winning EC of P. This follows from the same reasoning as for Rem. 12.10. ◁

Remark 12.12. Let U be a winning EC of P, strictly included in a losing EC V of P. Let $s \in V$ be a worst-case losing state (i.e., a state from which $\mathcal{P}_1$ cannot guarantee a strictly positive mean-payoff in the subgame $G \restriction V$, as defined at line 6 of the algorithm). Then we claim that $s \notin U$. Indeed, suppose the contrary. From s, $\mathcal{P}_2$ can enforce $\underline{\mathsf{MP}} \leq 0$ against any strategy $\lambda_1 \in \Lambda_1(P \restriction V)$, and a fortiori could do so against any strategy $\lambda_1 \in \Lambda_1(P \restriction U)$ (notice that only $\mathcal{P}_1$ can decide to leave U as U is an EC). Therefore, U would not be winning. ◁

Proof. We first show that the algorithm is *sound*, i.e., $\mathcal{M}_{\mathrm{W}} \subseteq \mathcal{U}_{\mathrm{W}}$. It is done by induction on the size of P. If P is empty, the claim is clear. Otherwise let $U \in \mathcal{M}_{\mathrm{W}}$. There are two cases.

1. U is equal to some U_i computed at line 4 and has never been removed from $\mathcal{M}_{\mathrm{W}}$. It means that L_i is empty, and by definition of L_i, that U_i is winning.

Also, U_i is trivially a *maximal* winning EC as it belongs to the maximal EC decomposition of P.

2. U has been added at line 8 as the result of some recursive call MWEC($P \restriction (U_i \setminus L_i)$) for some maximal EC U_i of P. Since $L_i \neq \emptyset$, the set $U_i \setminus L_i$ is strictly smaller than S, and by induction hypothesis, U is a winning EC of $P \restriction (U_i \setminus L_i)$. By Rem. 12.11, it is also a winning EC of P. It remains to show that U is a *maximal* winning EC of P. Suppose that it is not the case. Then there exists a strict superset U' of U which is a winning EC of P. Clearly, $U' \subseteq U_i$ since U_i is a maximal EC of P, and maximal ECs are pairwise disjoint. Morevoer, U' is a subset of $U_i \setminus L_i$ by Rem. 12.12. By Rem. 12.11, it is therefore a winning EC of $P \restriction (U_i \setminus L_i)$, which contradicts the maximality of U in $P \restriction (U_i \setminus L_i)$.

We now establish that the algorithm is *complete*, i.e., $\mathcal{U}_{\mathrm{W}} \subseteq \mathcal{M}_{\mathrm{W}}$. Again, it is proved by induction on the size of P. If P is empty, then the claim is obviously true. Now, suppose that P is non-empty, and let $U \in \mathcal{U}_{\mathrm{W}}$. There are two cases.

1. U is a maximal EC of P. In that case, it will be computed at line 4 and never removed from $\mathcal{M}_{\mathrm{W}}$ (because the set of losing states will be empty as U is winning).

2. U is not a maximal EC in P. Therefore there exists some maximal EC U_i of P, which is losing and strictly contains U. Let L_i be the non-empty set of worst-case losing states of U_i (as computed by line 6). We have to show that U is a maximal winning EC of $P \restriction (U_i \setminus L_i)$, in which case we could conclude by induction hypothesis, i.e., U would be returned by the recursive call MWEC($P \restriction (U_i \setminus L_i)$). By Rem. 12.12, U and L_i are disjoint, and therefore $U \subseteq (U_i \setminus L_i)$. By Rem. 12.10 and Rem. 12.11, U is a winning EC of $P \restriction (U_i \setminus L_i)$, since it is a winning EC of P. It remains to show that U is a *maximal* winning EC of $P \restriction (U_i \setminus L_i)$. Suppose that there exists a strict superset U' of U such that U' is a winning EC of $P \restriction (U_i \setminus L_i)$. By Rem. 12.11, U' would also be a winning EC of P, which contradicts that $U \in \mathcal{U}_{\mathrm{W}}$ by definition of $\mathcal{U}_{\mathrm{W}}$ as the set of maximal winning ECs. It implies that U is a maximal winning EC of $P \restriction (U_i \setminus L_i)$ and thus that $U \in \mathcal{M}_{\mathrm{W}}$.

Finally, consider the complexity class of this algorithm. Assume that we have an NP ∩ coNP oracle solving the worst-case threshold problem (Sect. 2.3.2) and called in line 6. The number of recursive calls to MWEC is linear in $|S|$, the number of states of P, because in the loop of line 5, the sets U_i are pairwise disjoint, and because each call is executed after eliminating at least one state. Moreover, the maximal EC decomposition of P can be computed in $O(|S|^2)$ [CH12]. Overall, we thus obtain that MWEC belongs to NP ∩ coNP. □

Consider execution of algorithm MWEC on the MDP described in Fig. 12.2. In its first call, it computes the maximal EC decomposition $\mathcal{M}_{\mathrm{W}} = \{U_1, U_2\}$. Now, for U_1, we have that L_1 is empty and thus U_1 remains in $\mathcal{M}_{\mathrm{W}}$. On the contrary, for U_2, we have that $L_2 = \{s_1, s_3, s_4\}$. Hence the algorithm suppresses U_2 from $\mathcal{M}_{\mathrm{W}}$ and recurses on the sub-MDP $P \restriction (U_2 \setminus L_2) = P \restriction \{s_5\}$. There, the maximal EC decomposition gives the unique EC U_3 which is winning since $L_3 = \emptyset$, and thus remains in $\mathcal{M}_{\mathrm{W}}$. The algorithm ends with $\mathcal{M}_{\mathrm{W}} = \{U_1, U_3\}$. Clearly we have that $\mathcal{M}_{\mathrm{W}} = \mathcal{U}_{\mathrm{W}}$ as proved before.

12.3.2 WECs are Almost-Surely Reached in the Long-Run

Recall that Lemma 12.6 states that under any arbitrary strategy $\lambda_1 \in \Lambda_1$, the set of infinitely visited states of the outcome of the MDP $P[\lambda_1] = G[\lambda_1, \lambda_2^{\mathsf{stoch}}]$ is almost-surely equal to an EC. In this section, we refine this result and show that under any *finite-memory* strategy $\lambda_1^f \in \Lambda_1^F$ *satisfying the BWC problem*, the set of infinitely visited states is almost-surely equal to a *winning* EC. In other words, the long-run probability of *negligible states*, defined as $S_{\mathsf{neg}} = S \setminus \bigcup_{U \in \mathcal{U}_{\mathrm{W}}} U = S \setminus \bigcup_{U \in \mathcal{W}} U$, is zero.

Lemma 12.13. *Let $G = (\mathcal{G}, S_1, S_2)$ be a two-player game, $\mathcal{G} = (S, E, w)$ its underlying graph, $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^M$ a memoryless stochastic model of $\mathcal{P}_2$, $P = G[\lambda_2^{\mathsf{stoch}}]$ the resulting MDP and $s_{\mathsf{init}} \in S$ the initial state. Let $\lambda_1^f \in \Lambda_1^F$ be a finite-memory strategy of $\mathcal{P}_1$ that satisfies the BWC problem for thresholds $(0, \nu) \in \mathbb{Q}^2$. Then, we have that*

$$\mathbb{P}_{s_{\mathsf{init}}}^{P[\lambda_1^f]}\left(\left\{\pi \in \mathsf{Outs}_{P[\lambda_1^f]}(s_{\mathsf{init}}) \mid \mathsf{Inf}(\pi) \in \mathcal{W}\right\}\right) = 1. \tag{12.4}$$

Proof. Let $\lambda_1^f \in \Lambda_1^F$ be a finite-memory strategy satisfying the BWC problem.

By Lemma 12.6, we have that eq. (12.4) is verified for $U \in \mathcal{E} = \mathcal{W} \cup \mathcal{L}$. It remains to show that the probability of having a losing EC, i.e., an EC in $\mathcal{L}$, is zero.

By contradiction, assume there exists some $U_{\mathrm{L}} \in \mathcal{L}$ such that

$$\mathbb{P}_{s_{\mathsf{init}}}^{P[\lambda_1^f]}\left(\left\{\pi \in \mathsf{Outs}_{P[\lambda_1^f]}(s_{\mathsf{init}}) \mid \mathsf{Inf}(\pi) = U_{\mathrm{L}}\right\}\right) > 0. \tag{12.5}$$

Since λ_1^f is finite-memory, we have that $M = P[\lambda_1^f]$ is a *finite* MC. Thus, we consider the bottom strongly-connected components (BSCCs) of M and eq. (12.5) implies that some outcomes of $\mathsf{Outs}_M(s_{\mathsf{init}})$ will be trapped in a BSCC corresponding to U_{L} (i.e., this BSCC is reachable with non-zero probability in M), and visit all its states infinitely often. Since U_{L} is losing, this BSCC induces plays where the mean-payoff is not strictly positive. Indeed, strategy $\lambda_2^{\mathsf{stoch}}$ of $\mathcal{P}_2$ suffices to produce consistent outcomes that are worst-case losing thanks to Def. 12.7 (as only the support matters for the worst-case requirement, not the exact probabilities). By prefix-independence of the mean-payoff value function, we obtain the existence of plays of M, starting in s_{init}, and inducing a mean-payoff that does not satisfy the worst-case threshold. This shows that λ_1^f is not winning for the BWC problem and by contradiction, concludes our proof. □

The direct consequence of this statement is that *edges involving negligible states do not contribute to the overall expectation* of finite-memory strategies satisfying the BWC problem. Based on this observation, we propose in Sect. 12.5 a modification of the MDP $P = G[\lambda_2^{\mathsf{stoch}}]$ that will help us synthesize satisfying strategies when they exist.

Remark 12.14. The proof of Lemma 12.13 relies on the finite memory of the strategy. Similar reasoning cannot be applied if the strategy of $\mathcal{P}_1$ uses infinite memory. Indeed, the Markov chain $M = P[\lambda_1^f]$ becomes infinite and we cannot base our analysis on BSCCs anymore (as they need not exist in general, outcomes cannot be trapped in a BSCC). As a matter of fact, we prove in Sect. 12.7 that infinite-memory strategies may benefit from negligible states forming losing ECs in some cases, by staying in them forever with a non-zero probability, and thus it is not possible to neglect them in the overall expectation. ◁

Illustration. Consider U_1 in Fig. 12.1. By Def. 12.7, this EC is losing as always taking the edge of weight -1 is a winning strategy for $\mathcal{P}_2$ in $G_\Delta \restriction U_1$.

The optimal expectation achievable in $P \downharpoonright U_1$ by $\mathcal{P}_1$ is 4: this is higher than what is achievable in both U_2 and U_3. Note that there exists no winning EC included in U_1. By Lemma 12.6, we know that any strategy of $\mathcal{P}_1$ will see its expectation bounded by the maximum between the optimal expectations of the ECs U_1, U_2 and U_3. Lemma 12.13 further refines this bound by restricting it to the maximum between the expectations of U_2 and U_3. Indeed, it states that $\mathcal{P}_1$ cannot benefit from the expected value of U_1 while using finite memory, as being trapped in U_1 with non-zero probability induces the existence of outcomes losing for the worst-case (here, outcomes that always take the -1 edge). Since U_1 neither helps for the worst-case nor for the expectation, there is no point in playing inside it and $\mathcal{P}_1$ may as well cross it directly and try to maximize its expectation using the winning ECs, U_2 and U_3. The set of negligible states in P is $S_{\mathsf{neg}} = S \setminus (U_2 \cup U_3) = \{s_1, s_2, s_3, s_4, s_5\}$.

In the game depicted in Fig. 12.2, we already observed that $\mathcal{E} = \{U_1, U_2, U_3\}$, $\mathcal{W} = \mathcal{U}_{\mathrm{w}} = \{U_1, U_3\}$ and $\mathcal{L} = \{U_2\}$. Consider the negligible state $s_1 \in S_{\mathsf{neg}} = U_2 \setminus U_3$. As a consequence of Lemma 12.13, we have that a finite-memory strategy of $\mathcal{P}_1$ may only take the edge (s_1, s_3) finitely often in order to ensure the worst-case requirement. Indeed, the losing outcome $(s_1 s_3 s_4)^\omega$ would exist (while of probability zero) if $\mathcal{P}_1$ were to play this edge infinitely often. Therefore, it is clear that $\mathcal{P}_1$ can only ensure that U_3 is reached with a probability arbitrarily close to one, and not equal to one, because at some point, he has to switch to edge (s_1, s_2) (after a bounded time since $\mathcal{P}_1$ uses a finite-memory strategy).

12.4 Inside Winning End-Components

12.4.1 WEC with Non-Zero Probabilities: Combined Strategy

In this section, we take a closer look at what happens inside a winning EC *where the stochastic model assigns non-zero probabilities* to all possible edges. We will show how to deal with edges of probability zero in Sect. 12.4.2. For the sake of readability, we make Assumption 12.15. Obviously, similar reasoning can be applied to all the such winning ECs in a larger game.

Assumption 12.15. Let $G = (\mathcal{G}, S_1, S_2)$ be a two-player game, $\mathcal{G} = (S, E, w)$ its underlying graph, $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^M$ a memoryless stochastic model of $\mathcal{P}_2$, and

$P = G[\lambda_2^{\mathsf{stoch}}] = (\mathcal{G}, S_1, S_\Delta = S_2, \Delta = \lambda_2^{\mathsf{stoch}})$ the resulting MDP. We assume that G is reduced to a unique maximal winning EC, i.e., $\mathcal{U}_{\mathrm{W}} = \{S\}$, and that $G_\Delta = G$, i.e., the set of edges of probability zero, $E \setminus E_\Delta$, is empty.

Our claim is that inside such a winning EC, $\mathcal{P}_1$ has a *finite-memory* strategy that simultaneously (a) ensures the worst-case requirement, and (b) yields an expected value which is ε-close to the maximal expectation of the EC. Consequently, we establish Theorem 12.16 and Corollary 12.17.

Theorem 12.16. *Let $G = (\mathcal{G}, S_1, S_2)$ be a two-player game reduced to a unique winning EC, $\mathcal{G} = (S, E, w)$ its underlying graph, $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^M$ a memoryless stochastic model of $\mathcal{P}_2$ such that $E_\Delta = E$, $P = G[\lambda_2^{\mathsf{stoch}}] = (\mathcal{G}, S_1, S_\Delta = S_2, \Delta = \lambda_2^{\mathsf{stoch}})$ the resulting MDP, $s_{\mathsf{init}} \in S$ an initial state and $\nu^* \in \mathbb{Q}$ the maximal expected value achievable by $\mathcal{P}_1$ in P. Then, for all $\varepsilon > 0$, there exists a finite-memory strategy of $\mathcal{P}_1$ that satisfies the BWC problem for the thresholds pair $(0, \nu^* - \varepsilon)$.*

The remainder of this section is dedicated to the proof of Thm. 12.16. It is a surprisingly positive result as it essentially states that $\mathcal{P}_1$ can *guarantee both* the worst-case and the expected value thresholds *without sacrifying any performance* (in terms of play values) except for some arbitrarily small ε. The key idea is to build a finite-memory strategy based on careful alternation between two memoryless strategies: one which is optimal for the worst-case, and one which is optimal for the expected value. The proof requires deep understanding of the limiting properties of Markov chains, such as the rate of convergence toward a stationary distribution. Nevertheless, we provide an intuitive sketch of the combined strategy and illustrate it on the running example in the following.

Corollary 12.17. *In a game G reduced to a winning EC, $\mathcal{P}_1$ has a strategy for the BWC problem for thresholds $(0, \nu)$ against a stochastic model $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^M$ such that $E_\Delta = E$ if and only if the optimal expected value in $G[\lambda_2^{\mathsf{stoch}}]$ is strictly greater than ν.*

Proof. Consider the left-to-right implication. Assume λ_1 is the BWC strategy. By definition, the optimal expected value ν^* is at least equal to $\mathbb{E}_{s_{\mathsf{init}}}^{P[\lambda_1]}(\underline{\mathsf{MP}})$, which is strictly greater than ν by hypothesis. Now consider the converse implication. Let ν^* be the optimal expected value. By hypothesis, $\nu^* > \nu$. By Thm. 12.16,

for any $\varepsilon > 0$, there exists a strategy $\lambda_1 \in \Lambda_1^F$ that satisfies the BWC problem for thresholds $(0, \nu^* - \varepsilon)$. In particular, it is possible to choose $\varepsilon \leq \nu^* - \nu$ to obtain the claim and conclude the proof. □

Individual requirements - memoryless strategies. First, consider the two requirements - worst-case and expected value - independently. By definition of winning ECs (eq. (12.2)) and the hypothesis that $G = G_\Delta$, $\mathcal{P}_1$ has a strategy to guarantee a strictly positive mean-payoff against any strategy of the adversary in the game. As discussed in Sect. 2.3.2, pure memoryless optimal strategies exist for $\mathcal{P}_1$. In the following, we denote $\lambda_1^{wc} \in \Lambda_1^{PM}$ such a strategy, and μ^* the optimal mean-payoff value, i.e., the minimal mean-payoff that λ_1^{wc} ensures against any strategy of $\mathcal{P}_2$. Hence, we have that

$$\begin{aligned}\mu^* &= \inf_{\lambda_2' \in \Lambda_2} \left\{ \underline{\mathsf{MP}}(\pi) \mid \pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1^{wc}, \lambda_2') \right\} \\ &= \sup_{\lambda_1 \in \Lambda_1} \inf_{\lambda_2' \in \Lambda_2} \left\{ \underline{\mathsf{MP}}(\pi) \mid \pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2') \right\} > 0,\end{aligned}$$

since we are in a winning EC for the worst-case threshold $\mu = 0$.

Similarly, we define a pure memoryless strategy maximizing the expected value in the MDP $P = G[\lambda_2^{\mathsf{stoch}}]$ induced by applying the memoryless stochastic model $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^M$ on G. We denote this strategy $\lambda_1^e \in \Lambda_1^{PM}$, and we write the associated expectation as $\nu^* = \mathbb{E}_{s_{\mathsf{init}}}^{P[\lambda_1^e]}(\underline{\mathsf{MP}})$. Notice that we manipulate equivalently strategies on the game and on the MDP thanks to their shared underlying graph (Sect. 12.2). The existence of a pure memoryless optimal strategy was discussed in Sect. 2.3.2.

However, we here require, *without loss of generality*, that λ_1^e is chosen[3] in order to satisfy an additional property: we want that the Markov chain $P[\lambda_1^e]$ be *unichain*, i.e., containing a unique recurrent class (i.e., a unique bottom strongly-connected component when considering edges which are assigned non-zero probability in the MC), and possibly some transient states. This will be useful later to apply needed technical results (Lemma 12.20). As evoked, it is always possible to choose such a strategy. Intuitively, $\mathcal{P}_2$ cannot force the existence of multiple recurrent classes in the MC as it would contradict the fact that we are inside an

[3]There may exist several pure memoryless optimal strategies, all yielding the same expected value, by definition of optimality.

EC (by definition, $\mathcal{P}_1$ must be able to force the visit of any state with probability one). Hence, it remains to argue that $\mathcal{P}_1$ has no interest in inducing multiple recurrent classes, as he cannot increase the expected value by doing so. This is clear since either the different recurrent classes yield the same expectation, in which case one suffices, or they yield different expectations, in which case an optimal strategy like λ_1^e will only use the one that produces the maximal expectation ($\mathcal{P}_1$ has the power to restrict the MC to the class of his choice by definition of EC).

In general, one cannot hope to satisfy the BWC problem by following only strategy λ_1^{wc} or only strategy λ_1^e, although they suffice when their respective requirements are considered independently. Fortunately, it is possible to build upon those two strategies in order to achieve simultaneous satisfaction with a combined finite-memory strategy.

Defining a combined strategy. Based on the existence of strategies λ_1^{wc} and λ_1^e, we define a pure finite-memory strategy $\lambda_1^{cmb} \in \Lambda_1^{PF}$ that carefully and dynamically alternates between the two memoryless strategies to ensure satisfaction of the BWC problem. Our strategy is *parameterized by two naturals* denoted by K and L.

Definition 12.18. In a game G satisfying Assumption 12.15, we define the *combined strategy* $\lambda_1^{cmb} \in \Lambda_1^{PF}$ as follows.

(a) Play λ_1^e for K steps[4] and memorize $\mathsf{Sum} \in \mathbb{Z}$, the sum of weights encountered during these K steps.

(b) If $\mathsf{Sum} > 0$, then go to *(a)*.

Else, play λ_1^{wc} during L steps then go to *(a)*.

We define *periods* as sequences played from the beginning of a phase of type *(a)* or *(b)* up to its end, i.e., the beginning of a new period. Intuitively, in a period of type *(a)*, the strategy mimics the optimal expectation strategy. By playing λ_1^e long enough, we can ensure that the mean-payoff obtained during the period is very close to ν^*, with probability close to one (Lemma 12.20). Still, we need to ensure the worst-case threshold in all cases. This may in general not

[4]By step we mean taking any edge in the game, be it from a state of $\mathcal{P}_1$ or $\mathcal{P}_2$.

be ensured by periods of type *(a)*. Hence, we keep a memory of the running sum of weights in the period. This requires a finite number of bits of memory as the sum takes values in $\{-K \cdot W, -K \cdot W + 1, \ldots, K \cdot W - 1, K \cdot W\}$. If $\mathsf{Sum} > 0$ after the K steps, then the mean-payoff over the period satisfies the worst-case requirement (eq. (11.1)) and the strategy can immediately start a new period of type *(a)*. Otherwise, it is necessary to compensate in order to satisfy the worst-case requirement. In that case, the strategy mimics the optimal worst-case strategy λ_1^{wc} for L steps. Such a strategy guarantees that cycles in the outcome have a strictly positive sum of weights since $\mu^* > 0$, as discussed before. As Sum is lower bounded after K steps, there exists a value of L such that the total sum of weights (and thus the mean-payoff) over periods *(a)* + *(b)* is strictly positive. Using the fact that all weights are integers, we further deduce that the sum over a period is at least equal to one. By the boundedness of the length of a period, we thus prove that the overall mean-payoff along a play stays *strictly* positive. Hence λ_1^{cmb} satisfies eq. (11.1).

While this sketch is sufficient to see that the worst-case requirement is satisfied by strategy λ_1^{cmb}, proving that we can choose K and L such that its expected value is ε-close to ν^* is more involved. Intuitively, periods of type *(b)* must not happen too frequently, nor be too long, in order to have a boundable impact on the overall expectation. The cornerstone to achieve such a moderate impact of periods of type *(b)* resides in the fact that a linear increase in K produces an exponential decrease in the need for a period of type *(b)* (Lemma 12.20) whereas it only requires a linear increase in L to ensure the worst-case requirement (see Def. 12.21 and Lemma 12.23). Note that the need for a decreasing contribution of periods of type *(b)* to the overall expectation explains why we need to track the current sum Sum and cannot settle for a simpler strategy that would play periods *(a)* and *(b)* in strict alternation (cf. Rem. 12.22).

In a nutshell, we claim that under Assumption 12.15, it is always possible to find values for constants K and L such that strategy λ_1^{cmb} satisfies the BWC problem for $(0, \nu^* - \varepsilon)$, as stated in Theorem 12.16. Before proving it, we illustrate the combined strategy and introduce some intermediary technical results. Notice that implementing λ_1^{cmb} only requires finite memory as strategies λ_1^{e} and λ_1^{wc} are memoryless, constants K and L have finite values, and Sum takes a finite number of values.

Illustration. Consider the subgame $G_\Delta \restriction U_3 = G \restriction U_3$ of the game in Fig. 12.1 and the initial state $s_{\mathsf{init}} = s_{10}$. Clearly, the worst-case requirement can be satisfied, that is why the EC is classified as winning. Indeed, always choosing to go to s_9 when in s_{10} is an optimal memoryless worst-case strategy λ_1^{wc} that guarantees a mean-payoff $\mu^* = 1$. The expectation of this strategy is $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^{wc}, \lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) = 1$. On the other hand, the strategy λ_1^e that always selects the edge going to s_{11} is optimal regarding the expected value criterion: it induces an expectation[5] $\nu^* = \Big(0 + \big(1/2 \cdot 9 + 1/2 \cdot (-1)\big)\Big)/2 = 2$ against the stochastic model $\lambda_2^{\mathsf{stoch}}$. However, it can only guarantee a mean-payoff of value $-1/2$ in the worst-case.

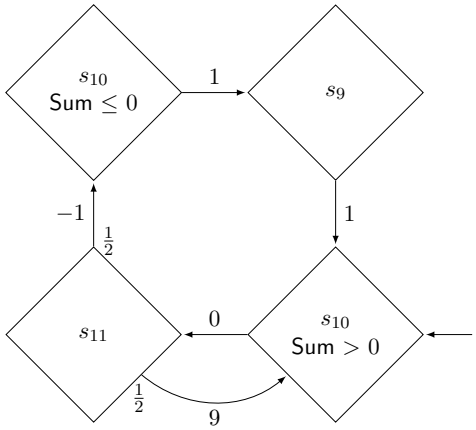


Figure 12.3: Markov chain $G[\lambda_1^{cmb}, \lambda_2^{\mathsf{stoch}}]$ induced by the combined strategy λ_1^{cmb} and the stochastic model $\lambda_2^{\mathsf{stoch}}$ over the winning EC U_3 of G.

By Theorem 12.16, we know that it is possible to find finite-memory strategies satisfying the BWC problem for any thresholds pair $(0, 2-\varepsilon)$, $\varepsilon > 0$. In particular, consider the thresholds pair $(0, 3/2)$. We build a combined strategy λ_1^{cmb} as described in Def. 12.18. Let $K = L = 2$: the strategy plays the edge (s_{10}, s_{11}) once, then if the edge of value 9 has been chosen by $\mathcal{P}_2$, it chooses (s_{10}, s_{11}) again;

[5] Given an irreducible MC $M = (\mathcal{G}, \delta)$, with $\mathcal{G} = (S, E, w)$, one can compute its limiting stationary distribution by finding the unique probability vector $\mathbf{v} \in [0,1]^{|S|}$ such that $\mathbf{v}\mathbf{P}_\delta = \mathbf{v}$, where $\mathbf{P}_\delta$ denotes the transition matrix derived from δ. The expected mean-payoff value can then be obtained by multiplying the row vector $\mathbf{v}$ by the column vector $\mathbf{e} \in \mathbb{R}^{|S|}$ that contains the respective expected weights over outgoing edges for each state. That is: $\forall\, s \in S$, $\mathbf{e}(s) = \sum_{s' \in S} \delta(s)(s') \cdot w((s,s'))$, and $\mathbb{E}_s^M = \mathbf{v} \cdot \mathbf{e}$.

otherwise it chooses the edge (s_{10}, s_9) once and then resumes choosing (s_{10}, s_{11}). This strategy satisfies the BWC problem. In the worst-case, $\mathcal{P}_2$ always chooses the -1 edge, but each time he does so, the -1 is followed by two $+1$ thanks to the cycle $s_{10}s_9s_{10}$. Strategy λ_1^{cmb} hence guarantees a mean-payoff equal to $(0-1+1+1)/4 = 1/4 > 0$ in the worst-case. For the expected value requirement, we can build the Markov chain $G[\lambda_1^{cmb}, \lambda_2^{\mathsf{stoch}}]$ (Fig. 12.3) and check that its expectation is $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^{cmb}, \lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) = 5/3 > 3/2$.

Remark 12.19. Memoryless strategies do not suffice for the BWC problem, even with randomization. Indeed, the edge (s_{10}, s_{11}) cannot be assigned a non-zero probability as it would endanger the worst-case requirement (since the outcome $(s_{10}s_{11})^\omega$ cycling on the edge of weight -1 would exist and have a negative mean-payoff). Hence, the only acceptable memoryless strategy is λ_1^{wc}, which has only an expectation of 1. ◁

Technical results. Before proving the correctness of strategy λ_1^{cmb}, we need to introduce an important property verified by the Markov chain $G[\lambda_1^e, \lambda_2^{\mathsf{stoch}}]$. It is well-known that in the long-run, the probability of outcomes that induce a mean-payoff equal to the expectation of the MC is one. Lemma 12.20 shows that, for sufficiently long prefixes, it is possible to bound the probability of having a mean-payoff which differs from the expected value by more than a given $\varepsilon > 0$. In particular, it implies that for sufficiently large values of K, this probability decreases exponentially with K. This will help us bound the impact of periods of type *(b)* on the overall expectation.

Lemma 12.20 (Follows from the extension of [GO02, Thm. 2] proposed in [Tra09]). *For all initial state s_{init}, for all $\varepsilon > 0$, for some constants $c_1, c_2 > 0$, there exists $K_0 \in \mathbb{N}$ such that, for all $K \geq K_0$,*

$$\mathbb{P}_{s_{\mathsf{init}}}^{G[\lambda_1^e, \lambda_2^{\mathsf{stoch}}]}\Big(\pi \in \mathsf{Plays}(G[\lambda_1^e, \lambda_2^{\mathsf{stoch}}]) \;\Big|\; \big|\underline{\mathsf{MP}}(\pi(K)) - \nu^*\big| \geq \varepsilon\Big) \leq \mathcal{F}(K, \varepsilon)$$
$$\text{with } \mathcal{F}(K, \varepsilon) = \frac{c_1}{e^{c_2 \cdot K \cdot \varepsilon^2}}.$$

In [GO02, Thm. 2], Glynn and Ormoneit present an extension of Hoeffding's inequality [Hoe63] for uniformly ergodic Markov chains. Straight application of this result in our setting is not possible, as the MC $G[\lambda_1^e, \lambda_2^{\mathsf{stoch}}]$ does not need to be aperiodic in general. Nevertheless, this result is extended to unichain MCs

(possibly periodic) in [Tra09]. Hence, Lemma 12.20 reformulates the latter result in our precise context. Notice that the MC $G[\lambda_1^e, \lambda_2^{\mathsf{stoch}}]$ is unichain thanks to the choice of λ_1^e, as presented earlier.

Proof. For the sake of completeness, we sketch the main steps proposed by Tracol [Tra09] to extend the results of [GO02]. Consider the MC $G[\lambda_1^e, \lambda_2^{\mathsf{stoch}}]$: it can be decomposed in a set of transient states and a unique recurrent class, i.e., a bottom strongly-connected component. Assume this BSCC is periodic, of period $d \in \mathbb{N}_0$. We can decompose it in d aperiodic classes on which we are able to apply the bound provided by [GO02, Thm. 2]. The key idea is then to obtain a unified bound by aggregating the bounds obtained for each aperiodic MC, given sufficiently large values of the constants.

A word on the constants of Lemma 12.20. Careful analysis of the proofs of [GO02, Thm. 2] and [Tra09, Prop. 2] reveals that c_1 is exponential in ε and polynomial in the characteristics of the MC, while c_2 is only polynomial in the characteristics of the MC. More importantly, constant K_0 is polynomial in the size of the MC and polynomial in the largest weight W (exponential in its encoding). □

Analysis of the combined strategy. Our goal is to show that for any $\varepsilon > 0$, there exist two naturals K and L such that λ_1^{cmb} proves the correctness of Theorem 12.16. First, we define L as a *linear* function of K. Note that the main purpose of K is to create periods of type *(a)* long enough to have an expected mean-payoff close to the optimal value achieved by λ_1^e, i.e., ν^*. The aim of L is for periods of type *(b)* to be long enough to compensate the possible negative effect of periods of type *(a)* and thus ensure the worst-case requirement. As stated before, L should not grow too quickly to preserve an overall mean-payoff which is mainly influenced by periods of type *(a)* (hence close to the optimal expectation ν^*).

Definition 12.21. Given a natural constant $K \in \mathbb{N}$, we define

$$L = \left\lfloor \frac{K \cdot W + |S| \cdot W + |S| \cdot \mu^*}{\mu^*} \right\rfloor + 1.$$

Remark 12.22. Obviously, L needs to be proportional to K to preserve the worst-case requirement as the amount to compensate is bounded by $K \cdot W$. Consequently, we can observe the need for the bookkeeping of Sum in strategy λ_1^{cmb} in contrast to a non-dynamic alternation scheme between periods of type *(a)* and *(b)*. Indeed, in a strategy following the latter scheme, the long-term expectation would be close to $\frac{K \cdot \nu^* + L \cdot \mu^*}{K+L}$. As L is not constant but proportional to K, one can easily see that this expression does not tend to ν^* when K tends to infinity (which is required when ε tends to zero according to Lemma 12.20). Thus, strict alternation does not suffice to satisfy the thresholds pair presented in Thm. 12.16. ◁

Under the value of L given in Def. 12.21, Lemma 12.23 states that the worst-case requirement is satisfied. The idea is to decompose any play into an infinite sequence of periods, each of them having a bounded length and ensuring a strictly positive sum of weights, thus yielding an overall strictly positive mean-payoff of the play.

Lemma 12.23. *For any $K \in \mathbb{N}$, the combined strategy $\lambda_1^{cmb} \in \Lambda_1^{PF}$ is such that*

$$\forall\, \lambda_2 \in \Lambda_2,\, \forall\, \pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1^{cmb}, \lambda_2),\, \underline{\mathsf{MP}}(\pi) > 0.$$

Proof. Let λ_2 be an arbitrary strategy of $\mathcal{P}_2$ and π any outcome taken in the set $\mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1^{cmb}, \lambda_2)$. By definition of λ_1^{cmb}, we decompose the play in a sequence of periods of type *(a)* and *(b)*. That is, $\pi = \rho_0\rho_1\rho_2\dots$ where, for all $i \geq 0$, ρ_i is a finite sequence of states that is either of length K if ρ_i is of type *(a)* or of length L if ρ_i is of type *(b)*. Moreover, ρ_0 is of type *(a)* and for all $i \geq 1$, ρ_i is of type *(b)* if and only if ρ_{i-1} is of type *(a)* and such that $\mathsf{Sum}(\rho_{i-1}) \leq 0$ (i.e., the sum of weights along the sequence is not strictly positive). We regroup each sequence of type *(b)* with its predecessor of type *(a)* and obtain $\pi = \rho'_0\rho'_1\rho'_2\dots$ such that all sequences are either of type *(a)* or of type *(a)* + *(b)*.

Consider any sequence ρ'_i of type *(a)*. Since it is not followed by a period of type *(b)*, we know that $\mathsf{Sum}(\rho'_i) > 0$. Now consider any sequence ρ'_i of type *(a)* + *(b)*. At the end of the period of type *(a)*, the sum is bounded from below by $-K \cdot W$. During the period of type *(b)*, an optimal *memoryless* worst-case strategy λ_1^{wc} is followed and consequently, all formed cycles have a mean-payoff at least equal to μ^*. Hence, the sum of weights over the period of type *(b)* is at

least $(L - |S|) \cdot \mu^* - |S| \cdot W$. By Def. 12.21, this induces that the overall sum over the sequence of type *(a)* + *(b)* is $\mathsf{Sum}(\rho'_i) > 0$.

In both cases, we thus have that $\mathsf{Sum}(\rho'_i) > 0$. But since weights are integers, this implies a stronger inequality: $\mathsf{Sum}(\rho'_i) \geq 1$. We go back to the play seen as an infinite sequence of states $\pi = s_0 s_1 s_2 \dots$. Thanks to the previous observations, we can now state that the total sum of weights up to any state s_t, $t \geq 0$, is bounded by

$$\mathsf{TP}(\pi(t)) \geq -\big[t \bmod (K+L)\big] \cdot W + \left\lfloor \frac{t}{K+L} \right\rfloor \cdot 1 \geq -(K+L) \cdot W + \frac{t}{K+L} - 1.$$

Hence the mean-payoff of the play π is

$$\underline{\mathsf{MP}}(\pi) \geq \liminf_{t \to \infty} \left[\frac{-(K+L) \cdot W}{t} + \frac{1}{K+L} - \frac{1}{t} \right] = \frac{1}{K+L} > 0,$$

which concludes the proof. □

It remains to show that for any $\varepsilon > 0$, there exists a value $K \in \mathbb{N}$ such that the expected value requirement (eq. (11.2)) is also satisfied by λ_1^{cmb}. This is proved in Lemma 12.24. Again, decomposition of plays into periods needs to be considered. Furthermore, the crux of the proof resides in the use of Lemma 12.20: it induces that when the length of periods of type *(a)* grows linearly, the probability of periods of type *(b)* decreases exponentially. Since the length of periods of type *(b)* only grows linearly in K, the overall impact of periods *(b)* in the expectation tends to zero when K tends to infinity. Hence, the expectation of λ_1^{cmb} tends to the optimal expectation ν^* and classical convergence analysis provides the result.

Lemma 12.24. *For any $\varepsilon > 0$, there exists $K \in \mathbb{N}$ such that*

$$\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^{cmb}, \lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) > \nu^* - \varepsilon.$$

Proof. For the proof, we assume that $\varepsilon \leq \nu^*$, otherwise the claim is obviously true for any $K \in \mathbb{N}$ since the mean-payoff of any outcome consistent with λ_1^{cmb} is strictly positive by application of Lemma 12.23. Now, for a given $K \in \mathbb{N}$, consider the corresponding finite MC $M(K) = G[\lambda_1^{cmb}, \lambda_2^{\mathsf{stoch}}]$ where λ_1^{cmb} is defined with the parameter K (i.e. periods of type *(a)* are of length K). To obtain the claim,

we prove that $\mathbb{E}_{s_{\mathsf{init}}}^{M(K)}(\underline{\mathsf{MP}}) \xrightarrow[K\to\infty]{} \nu^*$. Similarly to the proof of Lemma 12.23, any outcome of $M(K)$ is an outcome consistent with λ_1^{cmb} in G and thus can be decomposed in an infinite sequence of periods of types *(a)* and *(a)* + *(b)*.

To begin with, we will consider (i) the expectation over *one* period of type *(a)*, and (ii) the expectation over *one* period of type *(a)*+*(b)*, as well as the respective probabilities of seeing such periods whenever a new period begins. Note that we can do that because periods of type *(a)* and *(a)*+*(b)* are independent events by definition of λ_1^{cmb}. Hence, the probability that some period has a specific type, and the expected value of that period, are not influenced by what happened over previous periods.[6]

(i) Let us first consider periods of type *(a)*. By Def. 12.18, $\mathcal{P}_1$ follows the strategy λ_1^e during those periods. Recall that $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^e,\lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) = \nu^*$. Let us denote by $e_{(a)}$ the expected mean-payoff over a period of type *(a)*. By Lemma 12.20, for any $\varepsilon > 0$, there exists $K_0 \in \mathbb{N}$ such that for all $K \geq K_0$, this expected value is bounded from below by

$$e_{(a)} \geq (1 - \mathcal{F}(K,\varepsilon)) \cdot (\nu^* - \varepsilon) + \mathcal{F}(K,\varepsilon) \cdot x,$$

with x a lower bound on *any* consistent outcome. Since any period of type *(a)* is not followed by a period of type *(b)* (otherwise it would be considered as a period of type *(a)*+*(b)*), we know that the sum of weights along the K steps of the period is greater than or equal to 1, and therefore we can take $x \geq 1/K$ (cf. proof of Lemma 12.23).

(ii) Now, consider periods of type *(a)* + *(b)*. As shown in the proof of Lemma 12.23, the expected mean-payoff over such a period, denoted by $e_{(a)+(b)}$, is greater than or equal to $1/(K+L)$. Now consider the probability $p_{(a)+(b)}$ that a period is of type *(a)* + *(b)*. By definition of strategy λ_1^{cmb}, $p_{(a)+(b)}$ is equal to the probability of having a total sum of weights less than or equal to 0 after K steps of playing λ_1^e. Since we assume that $\varepsilon \leq \nu^*$ and since $\nu^* \geq \mu^* > 0$, we can bound from above this probability by the probability to have a mean-payoff less than $\nu^* - \varepsilon$ over the K steps. Repeating the argument of point (i) and applying Lemma 12.20, we deduce that $p_{(a)+(b)} \leq \mathcal{F}(K,\varepsilon)$ for all $K \geq K_0$.

[6] For the sake of simplicity, we omit that different periods do not necessarily start in the same state, as the resulting impact on the expectation is negligible for sufficiently long periods.

We now have sufficient arguments to study the overall expected value over m periods. Let $p_{(a)}$ denote the probability of periods of type *(a)*. Note that $p_{(a)} = 1 - p_{(a)+(b)}$. The expected number of periods of type *(a)* is $m \cdot p_{(a)}$. Similarly, $m \cdot p_{(a)+(b)}$ is the expected number of periods of type *(a)* + *(b)*. The expected sum of weights over the m periods is therefore $m \cdot p_{(a)} \cdot e_{(a)} \cdot K + m \cdot p_{(a)+(b)} \cdot e_{(a)+(b)} \cdot (K + L)$ since periods of type *(a)* have length K and periods of type *(a)* + *(b)* length $K + L$. The expected length of the m periods is $m \cdot p_{(a)} \cdot K + m \cdot p_{(a)+(b)} \cdot (K + L)$. Finally, we have that

$$\mathbb{E}^{M(K)}_{s_{\mathsf{init}},\, m\,\text{periods}}(\underline{\mathsf{MP}}) = \frac{m \cdot p_{(a)} \cdot e_{(a)} \cdot K + m \cdot p_{(a)+(b)} \cdot e_{(a)+(b)} \cdot (K + L)}{m \cdot p_{(a)} \cdot K + m \cdot p_{(a)+(b)} \cdot (K + L)}. \quad (12.6)$$

Clearly, this expression does not depend on the number of periods m. This is consistent with our analysis since we have established that periods are statistically independent. Also, note that this reasoning is only correct for complete periods. Nonetheless, any prefix $\pi(n)$, $n \geq 0$, of the play is composed of a sequence of complete periods followed by a suffix of length bounded by $(K + L)$ and of total sum of weights bounded by $-(K + L) \cdot W$ and $(K + L) \cdot W$. Hence, we frame the expected mean-payoff over the n first steps of a play by the two following inequalities, where $l = p_{(a)} \cdot K + p_{(a)+(b)} \cdot (K + L)$ denotes the expected length of a period:

$$\frac{\left\lfloor \frac{n}{l} \right\rfloor \cdot l \cdot \mathbb{E}^{M(K)}_{s_{\mathsf{init}},\, 1\,\text{period}}(\underline{\mathsf{MP}}) - (K + L) \cdot W}{n} \leq \mathbb{E}^{M(K)}_{s_{\mathsf{init}},\, n\,\text{steps}}(\underline{\mathsf{MP}}),$$

and

$$\mathbb{E}^{M(K)}_{s_{\mathsf{init}},\, n\,\text{steps}}(\underline{\mathsf{MP}}) \leq \frac{\left\lfloor \frac{n}{l} \right\rfloor \cdot l \cdot \mathbb{E}^{M(K)}_{s_{\mathsf{init}},\, 1\,\text{period}}(\underline{\mathsf{MP}}) + (K + L) \cdot W}{n}.$$

Naturally, the finite suffix proves to be negligible when n grows, hence

$$\mathbb{E}^{M(K)}_{s_{\mathsf{init}}}(\underline{\mathsf{MP}}) = \liminf_{n \to \infty} \left[\mathbb{E}^{M(K)}_{s_{\mathsf{init}},\, n\,\text{steps}}(\underline{\mathsf{MP}}) \right] = \mathbb{E}^{M(K)}_{s_{\mathsf{init}},\, 1\,\text{period}}(\underline{\mathsf{MP}}). \quad (12.7)$$

Observe that eq. (12.7) uses the equality between the expectation over the values of plays and the limit of the expectation over values of prefixes. This equality is verified for the mean-payoff value function but does not need to be true for arbitrary value functions.

Back to eq. (12.6), with $m = 1$, we use $e_{(a)+(b)} \geq 1/(K+L) > 0$ and assume $K > 0$ to obtain

$$\mathbb{E}^{M(K)}_{s_{\mathsf{init}}}(\underline{\mathsf{MP}}) \geq \frac{p_{(a)} \cdot e_{(a)}}{p_{(a)} + p_{(a)+(b)} \cdot \left(\frac{K+L}{K}\right)}.$$

Again, we assume K large enough to ensure $p_{(a)} > 0$ (such a K exists by consequence of Lemma 12.20) and get

$$\mathbb{E}^{M(K)}_{s_{\mathsf{init}}}(\underline{\mathsf{MP}}) \geq \frac{e_{(a)}}{1 + \left(\frac{p_{(a)+(b)}}{p_{(a)}}\right) \cdot \left(\frac{K+L}{K}\right)}.$$

By (i) and (ii), we have that $p_{(a)} \geq 1 - \mathcal{F}(K,\varepsilon)$, $p_{(a)+(b)} \leq \mathcal{F}(K,\varepsilon)$, and that $e_{(a)} \geq (1 - \mathcal{F}(K,\varepsilon)) \cdot (\nu^* - \varepsilon)$. We derive that

$$\mathbb{E}^{M(K)}_{s_{\mathsf{init}}}(\underline{\mathsf{MP}}) \geq \frac{(1 - \mathcal{F}(K,\varepsilon)) \cdot (\nu^* - \varepsilon)}{1 + \frac{\mathcal{F}(K,\varepsilon)\cdot(K+L)}{(1-\mathcal{F}(K,\varepsilon))\cdot K}}. \tag{12.8}$$

Recall that ultimately, we want to prove that $\mathbb{E}^{M(K)}_{s_{\mathsf{init}}}(\underline{\mathsf{MP}}) \xrightarrow[K\to\infty]{} \nu^*$. Consider what happens when $K \to \infty$ in eq. (12.8): notice that L is linear in K by Def. 12.21, hence $L \to \infty$, and that $\mathcal{F}(K,\varepsilon) \to 0$ by Lemma 12.20. This does not suffice to conclude on the possible convergence of the lower bound given in eq. (12.8). The crux of the argument is given by Lemma 12.20: $\mathcal{F}(K,\varepsilon)$ decreases exponentially for a linear increase in K. Thus, we have that $\mathcal{F}(K,\varepsilon)\cdot(K+L) \to 0$. Therefore,

$$\lim_{K\to\infty} \left[\frac{(1 - \mathcal{F}(K,\varepsilon)) \cdot (\nu^* - \varepsilon)}{1 + \frac{\mathcal{F}(K,\varepsilon)\cdot(K+L)}{(1-\mathcal{F}(K,\varepsilon))\cdot K}} \right] = \nu^* - \varepsilon. \tag{12.9}$$

Notice two facts. First, for any $K \in \mathbb{N}$, we have that

$$\mathbb{E}^{M(K)}_{s_{\mathsf{init}}}(\underline{\mathsf{MP}}) \leq \mathbb{E}^{G[\lambda_1^e, \lambda_2^{\mathsf{stoch}}]}_{s_{\mathsf{init}}}(\underline{\mathsf{MP}}) = \nu^*$$

by the optimality of λ_1^e. Second, eq. (12.9) is valid for any ε such that $\nu^* \geq \varepsilon > 0$. Hence we observe that the sequence of expected values $(\mathbb{E}^{M(K)}_{s_{\mathsf{init}}}(\underline{\mathsf{MP}}))_{K\geq 0}$ is

bounded from above and from below by two sequences converging to ν^*. Ergo

$$\mathbb{E}_{s_{\mathsf{init}}}^{M(K)}(\underline{\mathsf{MP}}) \xrightarrow[K\to\infty]{} \nu^*.$$

By definition of convergence, for any ε such that $\nu^* \geq \varepsilon > 0$, there exists $K \in \mathbb{N}$ such that $\mathbb{E}_{s_{\mathsf{init}}}^{M(K)}(\underline{\mathsf{MP}}) > \nu^* - \varepsilon$, which concludes our proof. □

Based on Lemma 12.23 and Lemma 12.24, Theorem 12.16 follows straightforwardly and concludes our analysis of winning ECs with no edges of probability zero. We will present how we can extend these results to arbitrary winning ECs in the next section.

12.4.2 Starting in a WEC: Witness-and-Secure Strategy

We now turn to winning ECs with potential existence of edges of probability zero, i.e., $E_\Delta \subseteq E$. We present in this section how to construct a finite-memory strategy that can benefit ε-closely from the maximal expectation achievable in such winning ECs when facing the stochastic model $\lambda_2^{\mathsf{stoch}}$, while guaranteeing satisfaction of the worst-case requirement even against *arbitrary* strategies of $\mathcal{P}_2$ (i.e., strategies that may use edges in $E \setminus E_\Delta$). It is crucial to notice that we now consider *a complete game*, i.e., not necessarily reduced to a single winning EC as in Sect. 12.4.1. Still, we assume that the play starts in a winning EC: consistent outcomes will stay in it when $\mathcal{P}_2$ follows $\lambda_2^{\mathsf{stoch}}$ (because $\mathcal{P}_1$ will have no interest to leave), but may exit the EC if $\mathcal{P}_2$ takes edges of probability zero, either by the action of $\mathcal{P}_2$ (recall there may exist edges that leave the EC in $E \setminus E_\Delta$) or the action of $\mathcal{P}_1$ (which may need to leave to guarantee a strictly positive mean-payoff).

Theorem 12.25. *Let $G = (\mathcal{G}, S_1, S_2)$ be a two-player game, $\mathcal{G} = (S, E, w)$ its underlying graph, $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^M$ a memoryless stochastic model of $\mathcal{P}_2$, $P = G[\lambda_2^{\mathsf{stoch}}] = (\mathcal{G}, S_1, S_\Delta = S_2, \Delta = \lambda_2^{\mathsf{stoch}})$ the resulting MDP, $U \in \mathcal{W}$ a winning EC, $s_{\mathsf{init}} \in U$ an initial state inside the EC, and $\nu^* \in \mathbb{Q}$ the maximal expected value achievable by $\mathcal{P}_1$ in $P \restriction U$. Then, for all $\varepsilon > 0$, there exists a finite-memory strategy of $\mathcal{P}_1$ that satisfies the BWC problem for the thresholds pair $(0, \nu^* - \varepsilon)$.*

We prove this theorem in the following. Let us first give some key intuition. With respect to the expected value requirement of eq. (11.2), the hypothesis is

that $\mathcal{P}_2$ *will* follow its stochastic model $\lambda_2^{\mathsf{stoch}}$. Hence, he will only play edges in E_Δ and the combined strategy proposed in Sect. 12.4.1 suffices to achieve the claimed expectation and guarantee the worst-case threshold against $\lambda_2^{\mathsf{stoch}}$ (basically, we can apply Thm. 12.16 on G_Δ). Now, we still have to account for arbitrary strategies of $\mathcal{P}_2$ to satisfy the worst-case requirement of eq. (11.1). Notice that the combined strategy suffices to ensure it against all strategies playing exclusively in E_Δ. So, it only remains to deal with strategies that chooses some edges in $E \setminus E_\Delta$. It is easy for $\mathcal{P}_1$ to *witness* if $\mathcal{P}_2$ chooses an edge outside E_Δ (as the stochastic model is assumed known by $\mathcal{P}_1$). If this happens, $\mathcal{P}_1$ can *secure* its mean-payoff value by switching from the combined strategy to a worst-case winning strategy, which exists in all states due to the preprocessing of the game (Sect. 12.2).

Basis strategies. We denote by $\lambda_1^{sec} \in \Lambda_1^{PM}(G)$ a pure memoryless worst-case winning strategy on G. This strategy exists in all states of the game due to the preprocessing, including states of the EC $U \in \mathcal{W}$. Still, it may require leaving the EC to ensure a strictly positive mean-payoff (because the definition of winning ECs only consider edges in E_Δ).

When using $\lambda_2^{\mathsf{stoch}}$, $\mathcal{P}_2$ cannot force leaving the winning EC $U \in \mathcal{W}$. By Thm. 12.16, $\mathcal{P}_1$ has a finite-memory strategy on $G_\Delta \restriction U$, denoted λ_1^{cmb}, that ensures eq. (11.2), and verifies eq. (11.1), *if we restrict* $\mathcal{P}_2$ to strategies in $\Lambda_2(G_\Delta)$.

Defining a witness-and-secure strategy. In order to prove Thm. 12.25, we define a pure finite-memory *witness-and-secure strategy* as follows.

Definition 12.26. In a game G such that $P = G[\lambda_2^{\mathsf{stoch}}]$, $U \in \mathcal{W}$ is a winning EC and $s_{\mathsf{init}} \in U$ is the initial state, we define the *witness-and-secure strategy* $\lambda_1^{wns} \in \Lambda_1^{PF}(G)$ as follows.

(i) Play the combined strategy $\lambda_1^{cmb} \in \Lambda_1^{PF}(G_\Delta \restriction U)$ as long as $\mathcal{P}_2$ picks edges in E_Δ.

(ii) As soon as $\mathcal{P}_2$ takes an edge in $E \setminus E_\Delta$,[7] play the worst-case winning strategy $\lambda_1^{sec} \in \Lambda_1^{PM}(G)$ forever.

[7] More complex switching schemes could be used, such as only switching if the edge taken is really dangerous (i.e., part of a non strictly positive cycle), switching after a bounded number of deviations from the support, etc. But this simple scheme proves to be sufficient to realize Thm. 12.25 and is easier to analyze.

This strategy uses the combined strategy λ_1^{cmb} presented in Def. 12.18. Hence it is similarly parameterized by two naturals K and L that respectively define the lengths of periods of type *(a)* and of type *(b)* in Def. 12.18. Again, we will show that for any $\varepsilon > 0$, we can find values for K and L such that Thm. 12.25 is verified. Notice that λ_1^{wns} is finite-memory since λ_1^{cmb} and λ_1^{sec} are too and watching for the appearance of an edge belonging to $E \setminus E_\Delta$ in the actions of the adversary only requires a finite amount of memory.

Intuitively, the witness-and-secure strategy acts as follows. As long as $\mathcal{P}_2$ conforms to E_Δ, playing in G is essentially the same as playing in G_Δ. Hence λ_1^{wns} prescribes acting like λ_1^{cmb}, which induces satisfaction of the BWC problem in the subgame $G_\Delta \restriction U$ by Thm. 12.16. Two requirements must be satisfied by strategy λ_1^{wns}: (a) the worst-case and (b) the expected value. First consider (a). Two situations may occur. Either the outcome is such that λ_1^{wns} always stays in phase (i) and strategy λ_1^{cmb} is used forever in $G_\Delta \restriction U$, in which case direct application of Thm. 12.16 suffices to prove that eq. (11.1) is satisfied. Or, the outcome is such that λ_1^{wns} switches to phase (ii), in which case satisfaction of the worst-case requirement follows by definition of λ_1^{sec}. Now consider (b). Notice that the only consistent outcomes always stay in phase (i), by definition of the set $\mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1^{wns}, \lambda_2^{\mathsf{stoch}})$ which does not allow for choices outside of E_Δ. Hence the overall expectation is equal to the one over outcomes staying in phase (i). By Def. 12.26, the latter is exactly the expectation of λ_1^{cmb}, which satisfies the threshold by Thm. 12.16. Thus, the existence of fitting values of K and L for Thm. 12.25 is granted.

Illustration. Consider the winning EC U_2 in the game depicted in Fig. 12.1 and the initial state $s_6 \in U_2$. Notice that $\mathcal{P}_1$ can ensure a strictly positive mean-payoff in the subgame $G_\Delta \restriction U_2$, but not in $G \restriction U_2$. Indeed, it is easy to see that by always choosing the -1 edges (which requires edge $(s_7, s_6) \in E_\Delta \setminus E$), $\mathcal{P}_2$ can ensure a negative mean-payoff whatever the strategy of $\mathcal{P}_1$. However, there exists a strategy that ensures eq. (11.1), i.e., yields a strictly positive mean-payoff against any strategy in $\Lambda_2(G)$, by leaving the EC. Let λ_1^{sec} be the memoryless strategy that takes the edge (s_6, s_9) and then cycle on $(s_{10}s_9)^\omega$ forever: it guarantees a mean-payoff of $1 > 0$.

For a moment, consider the EC U_2 in G_Δ. Graphically, it means that the -1 edge from s_7 to s_6 disappears. In the subgame $G_\Delta \restriction U_2$, there are two particular

memoryless strategies. The optimal worst-case strategy λ_1^{wc} guarantees a mean-payoff of $1/2 > 0$ by choosing to go to s_7. The optimal expectation strategy λ_1^{e} yields an expected mean-payoff of 3 by choosing to go to s_8 (notice this strategy yields the same expectation in $P_\Delta \restriction U_2$ and $P \restriction U_2$). Based on them, we build the combined strategy $\lambda_1^{cmb} \in \Lambda_1^{PF}(G_\Delta \restriction U_2)$ as defined in Def. 12.18 and by Thm. 12.16, for any $\varepsilon > 0$, there are values of K and L such that it satisfies the BWC problem for thresholds $(0, 3-\varepsilon)$ in $G_\Delta \restriction U_2$. For example, for $K = L = 2$, we have $\mathbb{E}_{s_6}^{(P_\Delta \restriction U_2)[\lambda_1^{cmb}]}(\underline{\mathsf{MP}}) = \mathbb{E}_{s_6}^{(P \restriction U_2)[\lambda_1^{cmb}]}(\underline{\mathsf{MP}}) = 13/6$.

We construct the witness-and-secure strategy $\lambda_1^{wns} \in \Lambda_1^{PF}(G)$ based on λ_1^{cmb} and λ_1^{sec} as described by Def. 12.26. In this case, that means playing as λ_1^{cmb} until the -1 edge from s_7 to s_6 is taken by $\mathcal{P}_2$. As previously sketched, such a strategy ensures a worst-case mean-payoff equal to $1 > 0$ thanks to λ_1^{sec} and yields an expectation $\mathbb{E}_{s_6}^{P[\lambda_1^{wns}]}(\underline{\mathsf{MP}}) = 13/6$ for $K = L = 2$.

Finally, notice that securing the mean-payoff by switching to phase (ii) of strategy λ_1^{wns} *is needed* to satisfy the worst-case requirement if $\mathcal{P}_2$ plays in $E \setminus E_\Delta$. Also, observe that it is still necessary to alternate according to λ_1^{cmb} in $G_\Delta \restriction U_2$ and that playing λ_1^{e} is not sufficient to ensure the worst-case (because $\mathcal{P}_1$ has to deal with the -1 edge from s_8 to s_6 that remains in E_Δ).

Analysis of the witness-and-secure strategy. We close our discussion of winning ECs with the formal proof of Thm. 12.25 through the use of the witness-and-secure strategy λ_1^{wns} (Def. 12.26).

Proof of Theorem 12.25. Assume an arbitrary $\varepsilon > 0$. Let $K, L \in \mathbb{N}$ be such that the combined strategy $\lambda_1^{cmb} \in \Lambda_1^{PF}(G_\Delta \restriction U)$, as defined in Def. 12.18, satisfies the BWC problem for thresholds $(0, \nu^* - \varepsilon)$ in $G_\Delta \restriction U$. The existence of such values is granted by Thm. 12.16. We build the finite-memory strategy $\lambda_1^{wns} \in \Lambda_1^{PF}(G)$ according to Def. 12.26 and claim it satisfies the BWC problem for the pair of thresholds $(0, \nu^* - \varepsilon)$ in G.

First, consider the worst-case requirement. Let $\lambda_2 \in \Lambda_2(G)$ be any strategy of $\mathcal{P}_2$ and $\pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1^{wns}, \lambda_2)$ be any outcome consistent with λ_1^{wns}. Two cases are possible. One, $\mathcal{P}_2$ keeps choosing edges in E_Δ forever. That is, for $\pi = s_0 s_1 s_2 \ldots$, for all $i \geq 0$ such that $s_i \in S_\Delta = S_2$, we have that $(s_i, s_{i+1}) \in E_\Delta$. Then, the play is constrained to $G_\Delta \restriction U$ and consistent with λ_1^{cmb}. Hence, it follows from Thm. 12.16 (and Lemma 12.23) that $\underline{\mathsf{MP}}(\pi) > 0$. Two, $\mathcal{P}_2$ chooses

some edges in $E \setminus E_\Delta$. That is, for $\pi = s_0 s_1 s_2 \dots$, there exists $i \geq 0$ such that $(s_i, s_{i+1}) \notin E_\Delta$. Let i_0 be the smallest index where it happens. By definition of λ_1^{wns}, we know that $\mathcal{P}_2$ switches to λ_1^{sec} at step i_0. Hence the suffix $\pi' = s_{i_0} s_{i_0+1} s_{i_0+2} \dots$ is consistent with λ_1^{sec}. Consequently, $\underline{\mathsf{MP}}(\pi') > 0$. By prefix-independence of the mean-payoff value function, we conclude that $\underline{\mathsf{MP}}(\pi) > 0$, which closes the case of the worst-case requirement.

Second, consider the expected value requirement. We claim that inequality $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^{wns}, \lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) > \nu^* - \varepsilon$ is verified. By definition, $\mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1^{wns}, \lambda_2^{\mathsf{stoch}}) = \mathsf{Outs}_P(s_{\mathsf{init}}, \lambda_1^{wns})$ only contains plays where $\mathcal{P}_2$ conforms to E_Δ at all times. Such plays never exit the EC and by Def. 12.26, we have $\mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1^{wns}, \lambda_2^{\mathsf{stoch}}) = \mathsf{Outs}_{G_\Delta}(s_{\mathsf{init}}, \lambda_1^{cmb}, \lambda_2^{\mathsf{stoch}})$. Also note that the probability measure of plays of those two sets is identical. Hence, we obtain

$$\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^{wns}, \lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) = \mathbb{E}_{s_{\mathsf{init}}}^{G_\Delta[\lambda_1^{cmb}, \lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) > \nu^* - \varepsilon$$

by Thm. 12.16 (and Lemma 12.24). This sets the case for the expectation and concludes our proof. □

12.5 Global Strategy: Favor Reaching the Highest Valued WECs

We now have all the elements needed to describe the final steps of algorithm BWC_MP (lines 12-17) and prove its correctness. In this section, we first describe (Def. 12.27) how to modify the weights of the MDP $P = G[\lambda_2^{\mathsf{stoch}}]$ such that a classical optimal expectation strategy in the modified MDP P' will naturally tries to reach *winning* ECs with the highest combined expectation. This step is a cornerstone of the *global strategy* $\lambda_1^{glb} \in \Lambda_1^{PF}(G)$ that we define next (Def. 12.29). This strategy is a by-product of algorithm BWC_MP.

We study the adequacy of algorithm BWC_MP through two lemmas. In Lemma 12.30, we prove its *correctness*, i.e., that if it returns Yes, then the global strategy λ_1^{glb} satisfies the BWC problem for the given thresholds. In Lemma 12.31, we show its *completeness*, i.e., that if it returns No, then there exists no finite-memory strategy that satisfies the BWC problem. Combining those two lemmas and the analysis of the preprocessing conducted in Sect. 12.2,

we conclude that algorithm BWC_MP is a valid algorithm to solve the BWC problem on any two-player game with the mean-payoff value function.

Modifying the MDP to naturally reach winning ECs. Our motivation is the creation of an MDP P' such that an optimal strategy in P' maximizes the expectation without using negligible states (as defined in Sect. 12.3.1, that is, with regard to G and P). Indeed, we know by Lemma 12.13 that winning ECs should be almost-surely eventually used in order to satisfy the worst-case requirement of the BWC problem. In particular, states in losing ECs and not in any winning sub-EC should be avoided in the long-run.

Definition 12.27. Given $G = (\mathcal{G}, S_1, S_2)$, $\mathcal{G} = (S, E, w)$ and $P = G[\lambda_2^{\mathsf{stoch}}]$, we define $G' = (\mathcal{G}', S_1, S_2)$, $\mathcal{G}' = (S, E, w')$ and $P' = G'[\lambda_2^{\mathsf{stoch}}]$ by modifying the weight function as follows:

$$\forall\, e = (s_1, s_2) \in E,\ w'(e) := \begin{cases} w(e) \text{ if } \exists\, U \in \mathcal{U}_{\mathrm{w}} \text{ s.t. } \{s_1, s_2\} \subseteq U, \\ 0 \text{ otherwise.} \end{cases}$$

Let $\lambda_1^e \in \Lambda_1^{PM}(P')$ be a pure memoryless strategy of $\mathcal{P}_1$ that maximizes the expected mean-payoff in P'. Such a strategy always exists (Sect. 2.3.2). Note that following λ_1^e does not suffice to satisfy the BWC problem in general (Rem. 12.19). This strategy will be part of the global strategy λ_1^{glb} (Def. 12.29): its role is to maximize the combined expectation of reachable winning ECs, while avoiding using negligible states, in particular states that only belong to losing ECs. The strategy to adopt inside winning ECs will be prescribed by another part of the global strategy, based on what we have established in Sect. 12.4.2. Observe that it suffices to consider the *maximal* winning ECs in order to maximize the expectation, as proved by Lemma 12.8.

Remark 12.28. Notice that λ_1^e is also well-defined in P and G thanks to the shared underlying graph. Also, recall that all states of G are worst-case winning due to the preprocessing. Let $\lambda_1^{wc} \in \Lambda_1^{PM}(G)$ be an optimal worst-case winning strategy. We observe that all states remain worst-case winning in G' for the reason that an optimal worst-case strategy only needs to visit edges involving negligible states finitely often. Indeed, either these negligible states do not belong to any EC, in which case $\mathcal{P}_1$ cannot rely on them to satisfy the worst-case requirement (as he cannot ensure that he will be able to see them infinitely often), or they belong

to losing ECs and no winning sub-EC, in which case $\mathcal{P}_1$ only needs to visit them a finite number of times (basically to get out of the set $(\bigcup_{U\in\mathcal{E}} U) \setminus (\bigcup_{U\in\mathcal{W}} U)$ if the play starts in it and reach the set $\bigcup_{U\in\mathcal{W}} U$). Hence, the guaranteed mean-payoff is not impacted by the changes described in Def. 12.27: it remains strictly positive in all states. By virtue of this, we deduce that

$$\forall\, s \in S,\ \mathbb{E}_{s_{\mathsf{init}}}^{P'[\lambda_1^e]}(\underline{\mathsf{MP}}) \geq \mathbb{E}_{s_{\mathsf{init}}}^{P'[\lambda_1^{wc}]}(\underline{\mathsf{MP}}) > 0,$$

and as such, that strategy λ_1^e will not prescribe staying in the set $(\bigcup_{U\in\mathcal{E}} U) \setminus (\bigcup_{U\in\mathcal{W}} U)$ forever. Indeed, it is always beneficial to exit it and obtain a strictly positive expectation instead of an expectation equal to zero (recall all edges involving negligible states are mapped to weight zero by Def. 12.27). ◁

Defining a global strategy. Based on the memoryless strategies λ_1^e and λ_1^{wc} in game G (as defined above), and the pure finite-memory witness-and-secure strategy λ_1^{wns} in winning ECs (as presented in Def. 12.26),[8] we build a *global strategy* λ_1^{glb} in G as follows. This strategy is parameterized by a natural constant $N \in \mathbb{N}$.

Definition 12.29. In a game G, we define the *global strategy* $\lambda_1^{glb} \in \Lambda_1^{PF}(G)$ as follows.

(a) Play $\lambda_1^e \in \Lambda_1^{PM}(G)$ for N steps.

(b) Let $s \in S$ be the reached state.

  (b.1) If $s \in U \in \mathcal{U}_{\mathrm{w}}$, play the corresponding strategy $\lambda_1^{wns} \in \Lambda_1^{PF}(G)$ forever.

  (b.2) Else play $\lambda_1^{wc} \in \Lambda_1^{PM}(G)$ forever.

Let us sketch this strategy. In phase (a), the optimal expectation strategy in P' is followed. It will drive the outcomes toward the ECs with the highest expected values. By taking N large enough, we can ensure that the probability of being in an EC will be arbitrarily close to one (by Lemma 12.6). As a result of the weights modification described in Def. 12.27, we can further ensure that the probability of being inside a *winning* EC will be arbitrarily close to one. Note

[8] Parameters K and L may vary depending on the actual corresponding EC.

that in phase $(b.1)$, the witness-and-secure strategy guarantees satisfaction of the worst-case requirement while granting an expectation arbitrarily close to the optimal expectation of the EC (as proved in Sect. 12.4.2). Also, in phase $(b.2)$, the mean-payoff of outcomes is strictly positive, and the probability of being in $(b.2)$ can be arbitrarily close to zero for large enough values of N. Overall, we obtain that λ_1^{glb} satisfies the worst-case requirement because the strategies played in the two terminal phases, $(b.1)$ and $(b.2)$, all guarantee its satisfaction and the mean-payoff is prefix-independent (hence it is not impacted by phase (a)). Furthermore, the expectation of λ_1^{glb} can be arbitrarily close to the maximal expectation ν^* achievable in P' (i.e., the one achieved by λ_1^e) by taking sufficiently large values for the constants K, L and N. Hence, if $\nu^* > \nu$, λ_1^{glb} is a proper BWC satisfying strategy for $\mathcal{P}_1$.

Finally, ν^* constitutes an upper bound to the expectation of any strategy of $\mathcal{P}_1$ in P'. By Lemma 12.13, it is also an upper bound on the expectation of any strategy that satisfies the worst-case requirement in the original game and MDP. It follows that if $\nu^* \leq \nu$, then there exists no finite-memory strategy that satisfies the BWC problem.

As the validity of the preprocessing was shown in Sect. 12.2, this let us conclude that algorithm BWC_MP is both correct *and* complete.

Illustration. Consider the game G depicted in Fig. 12.1 and the associated MDP $P = G[\lambda_2^{\mathsf{stoch}}]$. Following Lemma 12.6, analysis of the maximal ECs U_1, U_2 and U_3 reveals that the maximal expected mean-payoff achievable in P is 4. It is for example obtained by the memoryless strategy that chooses to go to s_2 from s_1 and to s_4 from s_3. Observe that playing in U_1 forever is needed to achieve this expectation. By Lemma 12.13, this should not be allowed as the worst-case cannot be ensured if it is. Indeed, $\mathcal{P}_2$ can produce worst-case losing outcomes by playing the -1 edge. Clearly, the maximal expected value that $\mathcal{P}_1$ can ensure while guaranteeing the worst-case requirement is thus bounded by the maximal expectation in P', i.e., by 3. Let λ_1^e denote an optimal memoryless expectation strategy in P' that tries to enter U_2 by playing (s_1, s_2) and (s_3, s_5), and then plays edge (s_6, s_8) forever (Fig. 12.4).

Observe that algorithm BWC_MP answers YES for any thresholds pair $(0, \nu)$ such that $\nu < 3$. For the sake of illustration, we construct the global strategy λ_1^{glb} as presented in Def. 12.29 with $N = 6$ and $K = L = 2$. For the first six steps,

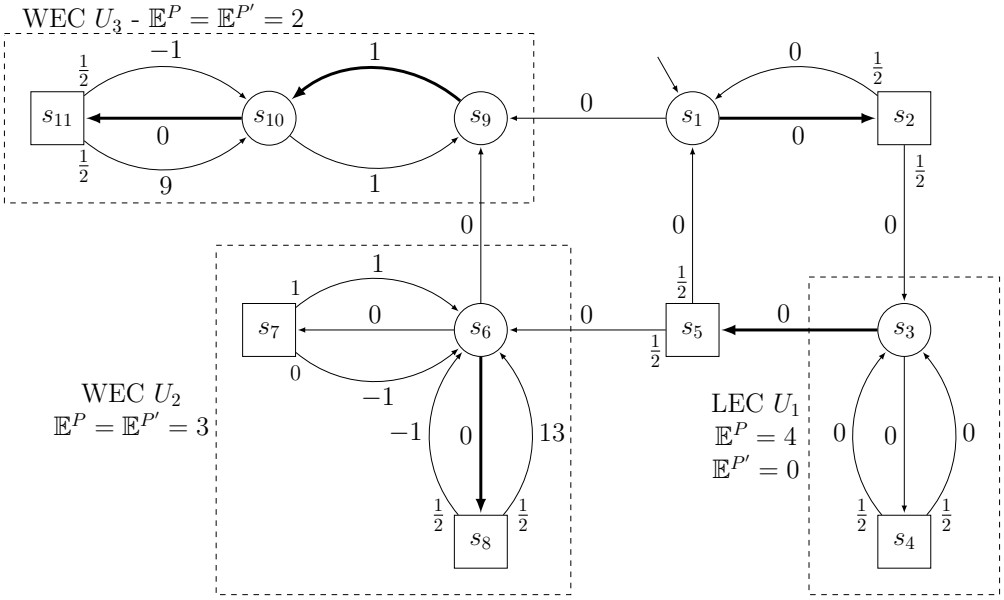


Figure 12.4: Putting all weights outside WECs to zero naturally drives the optimal expectation strategy in P', depicted by the thick edges, toward the highest valued WECs. ECs are annotated with their corresponding optimal expectations in the original MDP P and the modified MDP P'.

it behaves exactly as λ_1^e. Note that after the six steps, the probability of being in U_2 is $1/4 + 1/8 = 3/8$. Then, λ_1^{glb} switches to another strategy depending on the current state (λ_1^{wns} or λ_1^{wc}) and sticks to this strategy forever. Particularly, if the current state belongs to U_2, it switches to λ_1^{wns} as described in Def. 12.26 for $K = L = 2$, which guarantees the worst-case threshold and induces an expectation of 13/6 (Sect. 12.4.2). By definition of λ_1^{glb} on the sample game G, if the current state after six steps is not in U_2, then λ_1^{glb} switches to λ_1^{wc} which guarantees a mean-payoff of 1 by reaching state s_9 and then playing $(s_9 s_{10})^\omega$. Overall, the expected mean-payoff of λ_1^{glb} against $\lambda_2^{\mathsf{stoch}}$ is

$$\mathbb{E}_{s_1}^{G[\lambda_1^{glb}, \lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) \geq \frac{3}{8} \cdot \frac{13}{6} + \frac{5}{8} \cdot 1 = \frac{23}{16}.$$

Notice that by taking N, K and L large enough, it is possible to satisfy the BWC problem for any $\nu < 3$ with the strategy λ_1^{glb}. Also, observe that the winning EC U_2 is crucial to achieve expectations strictly greater than 2, which is the

upper bound when limited to EC U_3. For example, $N = 25$ and $K = L = 2$ implies an expectation strictly greater than 2 for the global strategy.

Lastly, note that in general, the maximal expectation achievable in P' (and thus in P when limited to strategies that respect the worst-case requirement) may depend on a combination of ECs instead of a unique one. This is transparent through the solving of the expected value problem in the MDP P'. Hence, the approach followed by algorithm BWC_MP is a way of solving a complex problem by breaking it into smaller pieces.

Correctness and completeness. We start by proving the correctness of the algorithm BWC_MP described in Alg. 12.1 and the soundness of the global strategy presented in Def. 12.29 to satisfy the BWC problem.

Lemma 12.30 (correctness). *If algorithm* BWC_MP *answers* YES*, then there exist values of the parameters such that the global strategy* $\lambda_1^{glb} \in \Lambda_1^{PF}$ *satisfies the BWC mean-payoff problem.*

Proof. We assume the answer returned by BWC_MP is YES and we prove the claim.

First, consider the worst-case requirement (eq. (11.1)). Let $\lambda_2 \in \Lambda_2$ be an arbitrary strategy of $\mathcal{P}_2$. Let N take an arbitrary value in $\mathbb{N}$, and for any winning EC $U \in \mathcal{U}_\mathrm{w}$, let K_U take an arbitrary value in $\mathbb{N}$ and L_U be defined according to Def. 12.21 with regard to K_U. Consider the outcomes consistent with λ_1^{glb} and λ_2. Our goal is to prove that for all outcomes $\pi \in \mathsf{Outs}_G(s_\mathsf{init}, \lambda_1^{glb}, \lambda_2)$, we have that $\underline{\mathsf{MP}}(\pi) > 0$. Let π be an arbitrary outcome in this set, $s = \mathsf{Last}(\pi(N))$ be the state reached after phase (a) of the global strategy, and π' be the suffix play such that $\pi = \pi(N) \cdot \pi'$. Two cases are possible. First, assume $s \in U$ for some maximal winning EC $U \in \mathcal{U}_\mathrm{w}$. Then, π' is consistent with the witness-and-secure strategy λ_1^{wns} (as presented in Sect. 12.4.2 for initial states in U). By Thm. 12.25, $\underline{\mathsf{MP}}(\pi') > 0$. Second, assume $s \notin \bigcup_{U \in \mathcal{U}_\mathrm{w}} U$, i.e., $s \in S_\mathsf{neg}$. Then, π' is consistent with the worst-case winning strategy λ_1^{wc} provided by the preprocessing, and we have that $\underline{\mathsf{MP}}(\pi') > 0$. By prefix-independence of the mean-payoff value function, we conclude that in both cases, $\underline{\mathsf{MP}}(\pi) = \underline{\mathsf{MP}}(\pi') > 0$, proving that strategy λ_1^{glb} ensures the worst-case requirement.

Second, consider the expected value requirement (eq. (11.2)). We need to prove that $\mathbb{E}_{s_\mathsf{init}}^{G[\lambda_1^{glb}, \lambda_2^\mathsf{stoch}]}(\underline{\mathsf{MP}}) > \nu$ for some well-chosen values of N and $K \in \mathbb{N}$.

Formally, the value K_U may be different in each winning EC $U \in \mathcal{U}_{\mathrm{W}}$, so we will take a uniform value K sufficiently large to ensure that it works for all ECs. Values $L_U(K)$ are defined according to Def. 12.21. Again noting that the weights encountered during phase (a) of strategy λ_1^{glb} have no impact on the mean-payoff of plays (because phase (a) is of finite duration and all weights are also finite), we formulate the expectation as

$$\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^{glb},\lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) = \sum_{U\in\mathcal{U}_{\mathrm{W}}} \big[p_N(U)\cdot e_K(U)\big] + \sum_{s\in S_{\mathsf{neg}}} \big[p_N(s)\cdot e_{\mathrm{WC}}(s)\big], \tag{12.10}$$

where $p_N(U)$ denotes the probability to be in a state belonging to the maximal winning EC $U \in \mathcal{U}_{\mathrm{W}}$ after N steps of following strategy λ_1^e (i.e., phase (a)); $e_K(U)$ denotes the expectation of plays starting in U and consistent with λ_1^{wns} for values K and $L_U(K)$ of the parameters (this expectation is identical for all initial states in the EC); $p_N(s)$ denotes the probability to be in a given negligible state $s \in S_{\mathsf{neg}}$ (i.e., outside of winning ECs) after phase (a); and $e_{\mathrm{WC}}(s)$ denotes the expectation over plays that start in such a state s and are consistent with the worst-case strategy λ_1^{wc}. Observe that $\sum_{s\in S_{\mathsf{neg}}} p_N(s) = 1 - \sum_{U\in\mathcal{U}_{\mathrm{W}}} p_N(U)$.

Similarly, we write the expectation of the optimal expectation strategy λ_1^e in P' as

$$\mathbb{E}_{s_{\mathsf{init}}}^{G'[\lambda_1^{e},\lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) = \sum_{U\in\mathcal{U}_{\mathrm{W}}} \big[p(U)\cdot e(U)\big], \tag{12.11}$$

where $p(U)$ and $e(U)$ denote the probability and the expectation of maximal winning ECs when strategy λ_1^e is followed forever. Note that eq. (12.11) depends uniquely on winning ECs by consequence of Rem. 12.28, and specifically maximal winning ECs by further application of Lemma 12.8. In addition, observe that

$$\mathbb{E}_{s_{\mathsf{init}}}^{G'[\lambda_1^{e},\lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) = \mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^{e},\lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) = \nu^*,$$

since the weight modification of Def. 12.27 does not alter winning ECs.

We claim that $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^{glb},\lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}})$ tends to ν^* when N and K tend to infinity. We study the terms of eq. (12.10). Note that e_{WC} takes a bounded value (in $]0, W]$ by definition of λ_1^{wc}). By application of the analysis developed in Lemma 12.24

and Theorem 12.25, we have that

$$\forall U \in \mathcal{U}_{\mathrm{W}},\ e_K(U) \xrightarrow[K\to\infty]{} e(U).$$

Furthermore, by definition of λ_1^e we have that

$$\forall U \in \mathcal{U}_{\mathrm{W}},\ p_N(U) \xrightarrow[N\to\infty]{} p(U),$$

and by definition of the modified weight function (Def. 12.27) and Rem. 12.28, that

$$\sum_{U\in\mathcal{U}_{\mathrm{W}}} p_N(U) \xrightarrow[N\to\infty]{} \sum_{U\in\mathcal{U}_{\mathrm{W}}} p(U) = 1.$$

Summing up, we obtain that $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^{glb},\lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) \xrightarrow[N,\,K\to\infty]{} \nu^*$. By convergence, for all $\varepsilon > 0$, there exist $N, K \in \mathbb{N}$ such that $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^{glb},\lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) \geq \nu^* - \varepsilon$. Since algorithm BWC_MP answered Yes, we have that $\nu^* > \nu$. Hence, there exist values $N, K \in \mathbb{N}$ such that $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^{glb},\lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) > \nu$. This concludes the proof. □

In order to prove that the algorithm solves the BWC problem (Def. 11.3) for the mean-payoff value function, we still need to establish its completeness: if the global strategy does not suffice to satisfy some thresholds pair, then no finite-memory strategy can do it.

Lemma 12.31 (completeness). *If algorithm* BWC_MP *answers* No*, then there exists no finite-memory strategy of* $\mathcal{P}_1$ *that satisfies the BWC mean-payoff problem.*

Proof. By contradiction, assume there exists $\lambda_1^f \in \Lambda_1^F$ that satisfies the BWC problem for thresholds $(0, \nu)$. We claim that algorithm BWC_MP answers Yes. First, notice that the algorithm cannot answer No at line 5 since λ_1^f satisfies the worst-case requirement from the initial state s_{init}. Hence it remains to prove that ν^*, as computed by the algorithm, is such that $\nu^* > \nu$. If it is the case, the algorithm will answer Yes, which proves our claim.

By hypothesis, strategy λ_1^f induces an expectation $\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^f,\lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) > \nu$. By

Lemma 12.13, we have that

$$\mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1^f,\lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) = \mathbb{E}_{s_{\mathsf{init}}}^{G'[\lambda_1^f,\lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) = \mathbb{E}_{s_{\mathsf{init}}}^{P'[\lambda_1^f]}(\underline{\mathsf{MP}}),$$

with G' the game obtained by the transformation defined in Def. 12.27. Moreover, by definition of the optimal expectation, we have that for all $\lambda_1 \in \Lambda_1$, $\nu^* \geq \mathbb{E}_{s_{\mathsf{init}}}^{P'[\lambda_1]}(\underline{\mathsf{MP}})$. In particular, this inequality is verified for strategy λ_1^f. Hence, we obtain that $\nu^* \geq \mathbb{E}_{s_{\mathsf{init}}}^{P'[\lambda_1^f]}(\underline{\mathsf{MP}}) > \nu$. Consequently, the answer of the algorithm is YES and the lemma is proved. □

In summary, correctness and completeness of algorithm BWC_MP as stated in Thm. 12.1 follows from the combination of Lemma 12.30, Lemma 12.31 and the validity of the preprocessing, as presented in Sect. 12.2. The complexity of the algorithm is discussed in the next section (Lemma 12.32), as well as matching lower bounds for the BWC problem (Lemma 12.34).

12.6 Complexity and Memory Bounds

12.6.1 Complexity: Algorithm and Lower Bound

We first prove that algorithm BWC_MP is in NP ∩ coNP (Lemma 12.32). The classical worst-case threshold problem also belongs to NP ∩ coNP and whether it is in P or not is a long-standing open problem (as discussed in Sect. 2.3.2). In Lemma 12.34, we establish that it reduces in polynomial time to the BWC problem. Given the outstanding nature of the worst-case threshold problem membership to P, algorithm BWC_MP can thus be considered optimal. Furthermore, we observe that if the worst-case threshold problem were proved to be solvable in deterministic polynomial time, then algorithm BWC_MP would also be in P (Rem. 12.33).

To prove the NP ∩ coNP-membership of the algorithm, we study each of its computing steps and observe that they all require polynomial time in the size of the input, except for a polynomial number of calls to an NP ∩ coNP algorithm solving the worst-case threshold problem. Rem. 12.33 is also induced by this observation.

Lemma 12.32. *Algorithm* BWC_MP *is in NP ∩ coNP.*

Proof. To begin with, note that the size of the *input* depends polynomially on (i) the number of states of the input game $|S^i|$, (ii) the number of edges of the input game $|E^i|$, (iii) the number of bits of the encoding of weights $V^i = \log_2 W^i$, (iv) the size of the memory of the SOMM $|\mathsf{Mem}|$, (v) the size of the supports and the length of the encoding of probabilities for the next-action function α_{n}, and (vi) the encoding of the thresholds $\mu, \nu \in \mathbb{Q}$.

To prove the NP ∩ coNP-membership of the algorithm, we review each step sequentially. Lines 1-2 and 4-10 are at most polynomial in the input. Line 3 consists in solving the worst-case threshold problem on the input game G^i: this can be done by calling an NP ∩ coNP algorithm (Sect. 2.3.2). Overall, the preprocessing is in NP ∩ coNP and produces a game G which size is bounded by $|G| \leq |G^i| \cdot |\mathcal{M}(\lambda_2^i)|$, using the natural definitions of those sizes as polynomial functions of the values described in points (i) to (vi).

For the main algorithm, the complexities are as follows. Line 11 is the call to the sub-algorithm MWEC(P_Δ), which has been proved to work in NP ∩ coNP in Lemma 12.9. Note that the size of $P = G[\lambda_2^{\mathsf{stoch}}]$ is polynomial in the input. The weights modification (line 12) requires linear time (polynomial in the input game) as do lines 14-17. Finally, computing the maximal expected value on P' (line 13) is polynomial in $|P'|$ via linear programming (Sect. 2.3.2), hence polynomial in the input size.

In conclusion, we observe that all operations of the algorithm are executed at most once, and each of them belongs to NP ∩ coNP, which proves the claim. □

Remark 12.33. Assume an algorithm PTIME_WC is established to solve the worst-case threshold problem in deterministic polynomial time. Then, the complexities of algorithm BWC_MP and sub-algorithm MWEC boil down to a polynomial number of polynomial-time operations and external calls, and it follows that BWC_MP is in P. ◁

Reduction of the worst-case threshold problem to the BWC one seems natural by eq. (11.1). Still, we need to pay attention to the strict inequality in the BWC problem definition: we use the existence of memoryless winning strategies for the worst-case problem and careful analysis of the domain of the mean-payoff values of outcomes to prove that it is not restrictive. The expected value part of the BWC problem can be defined arbitrarily under certain conditions.

Lemma 12.34. *The worst-case threshold problem on mean-payoff games reduces in polynomial time to the BWC mean-payoff problem.*

Proof. Given a game $G = (\mathcal{G}, S_1, S_2)$, its underlying graph $\mathcal{G} = (S, E, w)$, an initial state $s_{\mathsf{init}} \in S$, and the worst-case threshold $\mu = 0$ (without loss of generality), the worst-case threshold problem asks if the following proposition is true:

$$\exists\, \lambda_1 \in \Lambda_1, \forall\, \lambda_2 \in \Lambda_2, \forall\, \pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2), \underline{\mathsf{MP}}(\pi) \geq 0. \tag{12.12}$$

By the results presented in Sect. 2.3.2, it is equivalent to restrict both players to memoryless strategies:

$$\exists\, \lambda_1^{pm} \in \Lambda_1^{PM}, \forall\, \lambda_2^{pm} \in \Lambda_2^{PM}, \forall\, \pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1^{pm}, \lambda_2^{pm}), \underline{\mathsf{MP}}(\pi) \geq 0. \tag{12.13}$$

It is well-known that in this context, $\underline{\mathsf{MP}}(\pi) \geq 0 \Leftrightarrow \underline{\mathsf{MP}}(\pi) > -\frac{1}{|S|}$. Indeed, consider the following argument. First, the mean-payoff of any outcome $\pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1^{pm}, \lambda_2^{pm})$ can be trivially bounded by $-W \leq \underline{\mathsf{MP}}(\pi) \leq W$, with W the largest absolute value of any weight assigned by w to edges of G. Second, consider the decomposition of π into simple cycles (i.e., cycles with no repeated state except for the starting and ending state). Since weights are integers, any simple cycle has an associated mean-payoff belonging to $\{-W, \ldots, -\frac{1}{|S|}, 0, \frac{1}{|S|}, \ldots, W\}$. As both strategies are memoryless, any outcome $\pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1^{pm}, \lambda_2^{pm})$ will ultimately consist in a repeated simple cycle. Hence we have that $\underline{\mathsf{MP}}(\pi) \in \{-W, \ldots, -\frac{1}{|S|}, 0, \frac{1}{|S|}, \ldots, W\}$ and we observe that no value can be taken between $-\frac{1}{|S|}$ and 0.

Consequently, eq. (12.13) is equivalent to

$$\exists\, \lambda_1^{pm} \in \Lambda_1^{PM}, \forall\, \lambda_2^{pm} \in \Lambda_2^{PM}, \forall\, \pi \in \mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1^{pm}, \lambda_2^{pm}), \underline{\mathsf{MP}}(\pi) > -\frac{1}{|S|}.$$

To formulate this last equation in terms of a BWC problem, we have to define an expected value threshold $\nu \in \mathbb{Q}$ and a stochastic model $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^F$. Since all plays $\pi \in \mathsf{Plays}(\mathcal{G})$ satisfy $\underline{\mathsf{MP}}(\pi) \geq -W$ by definition of the weight function, we trivially have that

$$\forall\, \lambda_1 \in \Lambda_1, \forall\, \lambda_2^{\mathsf{stoch}} \in \Lambda_2^F, \mathbb{E}_{s_{\mathsf{init}}}^{G[\lambda_1, \lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) \geq -W.$$

Hence, it suffices to fix an arbitrary stochastic model $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^F$ and an arbitrary expectation threshold $\nu < -W$ to obtain that eq. (12.12) is satisfied *if and only if* $\mathcal{P}_1$ has a strategy to satisfy the BWC problem for thresholds $(-\frac{1}{|S|}, \nu)$ against the stochastic model $\lambda_2^{\mathsf{stoch}}$.

Notice this reduction is polynomial because we can choose a simple stochastic model (e.g., a memoryless strategy requires a SOMM of size linear in the size of the game) and a value of ν that will not require a super-polynomial growth of the encodings (e.g., $\nu = -W - 1$). □

Our complexity results are summed up in Thm. 12.1.

12.6.2 Memory Requirements

Across the previous sections, we have studied the complexity of deciding the BWC problem, i.e., deciding the existence of a *finite-memory* strategy of $\mathcal{P}_1$ satisfying Def. 11.3 for the mean-payoff value function. Now, we focus on the *size of the memory* used by such a strategy. In Thm. 12.35, we give an upper bound for the memory of the global strategy described in Def. 12.29 (which has been shown to suffice if satisfaction of the BWC problem is possible). This is obtained through careful analysis of the structure of involved strategies (global, witness-and-secure, combined). All of them are based on alternation between well-chosen pure memoryless strategies, based on parameters N, K and $L \in \mathbb{N}$. We prove that these values only need to be polynomial in the size of the game and the stochastic model, and in the values of weights and thresholds, granting the claim.

Furthermore, we prove this upper bound to be tight in the sense that polynomial memory in the values of weights is needed in general. To establish this result, we provide a family of games $(G(X))_{X \in \mathbb{N}_0}$, reduced to a winning EC and where all possible edges are assigned non-zero probability by the stochastic model (i.e., verifying Assumption 12.15). This family is presented in Fig. 12.5. By choosing the worst-case threshold to be $\mu = 0$ and the expectation threshold to be $\nu \in]1, 5/4[$, we ensure that the BWC problem is satisfiable and that it cannot be achieved by the memoryless strategy that always chooses edge (s_1, s_2). Intuitively, it is thus mandatory to choose (s_1, s_3) infinitely often in order to achieve the BWC problem. Moreover, after some point, everytime this edge is

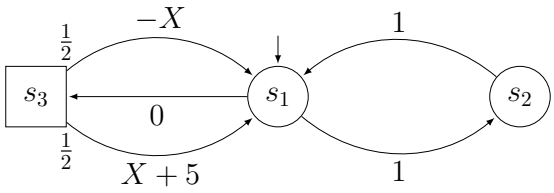


Figure 12.5: Family of games $(G(X))_{X\in\mathbb{N}_0}$ requiring polynomial memory in $W = X + 5$ to satisfy the BWC mean-payoff problem for thresholds $(0, \nu \in]1, 5/4[)$.

chosen, a satisfying strategy must be able to *eventually* counteract the potential negative weight $-X$ by taking edge (s_1, s_2) for $\lfloor X/2 \rfloor + 1$ times. This proves that polynomial memory in W is needed.

Theorem 12.35. *Memory of pseudo-polynomial size may be necessary and is always sufficient to satisfy the BWC problem for the mean-payoff: polynomial in the size of the game and the stochastic model, and polynomial in the weight and threshold values.*

Proof. We first consider the upper bound on memory, derived by analysis of the global strategy λ_1^{glb} (Def. 12.29). Observe that it follows the memoryless strategy λ_1^e for N steps before switching to phase *(b)*. The correctness of the strategy (Lemma 12.30) relies on the existence of a value N such that the probability of being in an EC after N steps is *high enough*. We argue that N does not need to be exponentially large.

Consider the probability to be outside of winning ECs after N steps. Applying classical results on Markov chains, we obtain that this probability decreases exponentially fast when N grows. Indeed, to prove it, it suffices to consider the chain $G[\lambda_1^e, \lambda_2^{\mathsf{stoch}}]$, replace BSCCs by absorbing states and observe that the probability of absorption tends toward one exponentially fast [GS97]. Now, consider the expectation of the global strategy for given constants N and K, as given in eq. (12.10). Let $\nu < \nu^* - \varepsilon$ be the expected value threshold considered in the BWC problem (as before, we assume $\mu < \nu < \nu^*$ otherwise the problem is trivial). Assume that K is sufficiently large to have $\sum_{U\in\mathcal{U}_w} p(U) \cdot e_K(U) > \nu^* - \varepsilon'$, with $\varepsilon' < \varepsilon$. We want to establish how large N needs to be to ensure an overall expectation strictly greater than ν. Since $e_{\mathrm{WC}}(s)$ can be trivially lower bounded by zero for all $s \in S_{\mathsf{neg}}$, it is clear that to obtain the needed property, we need to

have values $p_N(U)$ growing polynomially with ε and ν^*. As the growth of $p_N(U)$ is exponential in the growth of the value N, we obtain that a logarithmic value of N, hence *polynomial in the encoding*, suffices to achieve the desired expected value.

Similarly, we study the strategies followed in phase *(b)* of the global strategy (Def. 12.29). The case *(b.2)* is the easiest: the worst-case strategy λ_1^{wc} is memoryless. In case *(b.1)*, the witness-and-secure strategy λ_1^{wns} is used. By Def. 12.26, this strategy needs polynomial memory to witness the use of edges in $E \setminus E_\Delta$ and to implement the memoryless secure strategy λ_1^{sec}. It also needs to implement the combined strategy λ_1^{cmb}, based on alternation between memoryless strategies. The size of the memory of λ_1^{cmb} is polynomial in K, L and the largest absolute value taken by Sum, as well as in the size of the game. The proof of Lemma 12.20 guarantees that for constant K, a value polynomial in the size of the input game and the stochastic model, as well as in the values of weights and thresholds, suffices. By Def. 12.21, an identical situation is verified for L. Finally, the running sum Sum takes values in $\{-K \cdot W, \ldots, K \cdot W\}$, hence it also verifies such bounds. Overall, the memory needed by the combined strategy is polynomial in the size of the input game and the stochastic model, and in the weight and threshold values.

Aggregating all these bounds, we conclude that the global strategy also requires memory at most polynomial in the size of the input game and the stochastic model, and in the values, thus proving the upper bound.

It remains to show that pseudo-polynomial memory is really necessary in general. In order to achieve this, we introduce a family of games, $(G(X))_{X \in \mathbb{N}_0}$, such that winning the BWC problem on $G(X)$ requires memory polynomial in the largest weight $W = X + 5$. This family is presented in Fig. 12.5. Let the worst-case threshold be $\mu = 0$ and the expectation threshold be an arbitrary value $\nu \in]1, 5/4[$. Thanks to Thm. 12.16, the BWC problem is satisfiable, because $G(X)$ is reduced to a winning EC with no edge of probability zero, and the optimal expectation is $5/4 > \nu$ (expectation achieved by the memoryless strategy that always chooses edge (s_1, s_3)). Notice that it cannot be achieved by the memoryless strategy that always chooses edge (s_1, s_2) since this strategy induces a mean-payoff equal to $1 < \nu$. Hence it is mandatory to choose (s_1, s_3) infinitely often in order to achieve the expected value requirement (eq. (11.2)).

Let $\lambda_1 \in \Lambda_1^F$ be a finite-memory strategy of $\mathcal{P}_1$ that satisfies the BWC problem. Observe that it may as well be pure, i.e., $\lambda_1 \in \Lambda_1^{PF}$ as choosing edge (s_1, s_3) with a non-zero probability recurrently yields consistent outcomes that do not satisfy the worst-case requirement (eq. (11.1)). Also observe that anytime edge (s_1, s_3) is chosen, there is a probability $1/2$ that the edge of weight $-X$ is taken to come back. Hence, from some point on, every appearence of this edge of weight $-X$ must be *eventually* counteracted in order to preserve the worst-case requirement. A finite number of non-compensated occurences is not a problem thanks to the prefix-independence of the mean-payoff value function. Looking at the involved weights, it is clear that taking the edge (s_1, s_2) for $(\lfloor X/2 \rfloor + 1)$ times is necessary to counteract the negative edge of weight $-X$. Hence, memory polynomial in X (hence in W) is needed to ensure both the worst-case and the expected value requirements for the given thresholds. This concludes our proof. □

12.7 Infinite Memory

We close our study of the BWC problem for the mean-payoff value function by considering what happens when $\mathcal{P}_1$ is allowed to use infinite-memory strategies. Specifically, we show that in this context, infinite-memory strategies are in general strictly more powerful than finite-memory ones: they can exploit losing ECs to benefit from their possibly higher optimal expected value; and even inside a single winning EC, they can be optimal with regard to the expectation whereas finite-memory ones are limited to ε-optimality. Nonetheless, as discussed before, such strategies are ill-suited for the synthesis of implementable controllers for real-world applications, hence our focus on finite memory in the previous results.

Losing end-components may still be useful. Let us consider the game depicted in Fig. 12.6, together with a memoryless stochastic model of the adversary $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^M$, modeled by the probabilities $1/10$ and $9/10$ on the edges leaving s_1. The MDP $P = G[\lambda_2^{\mathsf{stoch}}]$ can be decomposed into two end-components U_1 and U_2, as depicted by the dashed lines. Assume the worst-case threshold is $\mu = -3/2$ (notice for once we take it different than zero), then U_1 is losing (because $\mathcal{P}_2$ can induce an outcome of mean-payoff value $-4/2 \leq -3/2$, and he can do that by

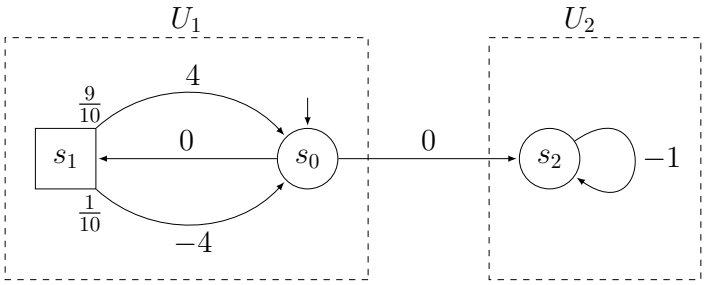


Figure 12.6: Infinite-memory strategies may use losing ECs forever with a non-zero probability in order to increase the expected value.

choosing edges in E_Δ) and U_2 is winning (as the only outcome yields mean-payoff $-1 > -3/2$), following Def. 12.7.

As shown in Lemma 12.13, any finite-memory strategy of $\mathcal{P}_1$ which ensures a mean-payoff strictly greater than μ, leaves U_1 with probability one against $\lambda_2^{\mathsf{stoch}}$, because states of U_1 are classified as negligible. Therefore, in order to satisfy the worst-case requirement of the BWC problem, the expected mean-payoff of any finite-memory strategy of $\mathcal{P}_1$ is $\nu_2^* = -1$, i.e., the expectation obtained in U_2 by the only possible outcome. Notice however that if we forget about the worst-case requirement, the maximal expectation that $\mathcal{P}_1$ could achieve in U_1 is $\nu_1^* = \frac{1}{2} \cdot (\frac{9}{10} \cdot 4 + \frac{1}{10} \cdot (-4)) = \frac{8}{5}$.

We now show that $\mathcal{P}_1$ can ensure the worst-case requirement and obtain an expected value strictly greater than -1, if he is allowed to use infinite memory. We define a pure infinite-memory strategy λ_1 for $\mathcal{P}_1$ as follows: λ_1 stores the running sum along the prefix played so far, and chooses to move from s_0 to s_1 as long as this sum is strictly greater than zero (except in the first round where it moves directly to s_1). First, notice that this strategy trivially guarantees a mean-payoff greater than or equal to $-1 > \mu$. Indeed, either the running sum always stays strictly positive, implying that the mean-payoff is at least zero, or the running sum gets negative or null at some point, in which case the strategy switches to U_2 and the mean-payoff takes value -1. Second, let us compute the expected mean-payoff of this strategy against $\lambda_2^{\mathsf{stoch}}$. Let p_{switch} denote the probability to switch to U_2 along a play, as prescribed by the strategy λ_1. By definition, it is equal to the probability, when playing inside the EC U_1 and starting from an initial credit of zero in state s_0, to come back to s_0 with a

credit less than or equal to zero after an arbitrary number of steps. Formally, let M be the MC induced by the subgame $G \restriction U_1$ and $\lambda_2^{\mathsf{stoch}}$ (note that $\mathcal{P}_1$ has no choice in it). We have

$$p_{\mathsf{switch}} = \mathbb{P}_{s_0}^{M}\big(\{\pi \in \mathsf{Outs}_M(s_0) \mid \exists\, i > 0,\, \mathsf{Last}(\pi(i)) = s_0 \wedge \mathsf{TP}(\pi(i)) \leq 0\}\big).$$

Determining the probability p_{switch} of the running sum hitting zero is equivalent to a well-studied problem on Markov chains, known as the *gambler's ruin problem* (see for instance [GS97]). Applying results on this problem, we obtain that, if the probabilities $9/10$ and $1/10$ are respectively replaced by arbitrary probabilities p and q such that $p > q$, the probability that the gambler is eventually ruined is $\frac{q}{p}$. In our example, this implies that $p_{\mathsf{switch}} = 1/9$.

We are now able to use this result to provide a lower bound for the expected value of the strategy λ_1. Consider the set of outcomes $\mathsf{Outs}_G(s_0, \lambda_1, \lambda_2^{\mathsf{stoch}})$: it can be partitioned into the set of plays that stay in U_1, for which the mean-payoff is trivially bounded by zero as discussed before; and the set of plays that reach U_2, for which the mean-payoff is equal to -1. Hence, the overall expectation respects the following inequality:

$$\mathbb{E}_{s_0}^{G[\lambda_1, \lambda_2^{\mathsf{stoch}}]}(\underline{\mathsf{MP}}) \geq p_{\mathsf{switch}} \cdot (-1) + (1 - p_{\mathsf{switch}}) \cdot 0 = -\frac{1}{9} > -1.$$

Clearly, we see that strategy λ_1 yields an expectation at least equal to $-1/9$, hence strictly greater than the expectation achievable by any finite-memory strategy satisfying the BWC problem, which we have shown to be equal to -1.

Intuitively, the added power given by infinite memory comes from the possibility to memorize an unbounded running sum of weights, whereas finite memory implies an upper bound on such a sum. In the first case, $\mathcal{P}_1$ will be able to properly acknowledge that some plays see their running sum diverging without ever dropping to zero (the set of such plays has a strictly positive probability in our example), which lets him benefit from the added expectation without endangering the worst-case requirement. In the second case, $\mathcal{P}_1$ sees all running sums as upper bounded by some value $X \in \mathbb{N}$ due to its limited memory. As such, when he sees a sequence of weights whose total sum is $-X$, an event that occurs almost-surely infinitely often when an outcome π is such that $\mathsf{Inf}(\pi) = U$ for some EC $U \in \mathcal{L} = \mathcal{E} \setminus \mathcal{W}$, $\mathcal{P}_1$ will *believe* its running sum hits zero, *whether it*

really does or not. Consequently, he has to leave U_1 to ensure the worst-case requirement at some point.

Optimal expected values can be reached in winning end-components. Consider a setting satisfying Assumption 12.15: a game G reduced to a winning EC such that $E_\Delta = E$. Let the worst-case threshold be $\mu = 0$, as usual. In Sect. 12.4.1, we have seen that, for all $\varepsilon > 0$, it is possible to combine a worst-case strategy λ_1^{wc} with an optimal expectation strategy λ_1^e into a finite-memory strategy λ_1^{cmb} that ensures satisfaction of the BWC problem for thresholds $(0, \nu^* - \varepsilon)$, where ν^* is the maximal expected value in $P = G[\lambda_2^{\mathsf{stoch}}]$. Observe that in general, it is not possible to construct a finite-memory strategy λ_1^{cmb} that ensures the worst-case while inducing an expected value exactly equal to ν^* against the stochastic model. See for example the game $G \downharpoonright U_3$ in Fig. 12.1: clearly, $\mathcal{P}_1$ has to use (s_{10}, s_9) infinitely often to ensure the worst-case, and when using finite memory, the contribution (in terms of proportion of cycles played) of the corresponding cycle in the overall expectation can be lower bounded by a strictly positive probability, hence inducing an expected value strictly lower than $\nu^* = 2$.

Nonetheless, it is possible to build an infinite-memory strategy, denoted λ_1^{inf}, that exactly achieves this expectation while verifying the worst-case threshold. It is in essence similar to the finite-memory combined strategy (Def. 12.18). Consider the following argument. Observe that in the analysis of the combined strategy (Sect. 12.4.1), we show that when parameters K and $L(K)$ tend to infinity, the expectation induced by λ_1^{cmb} tends to ν^*. Moreover, the worst-case is always ensured by choice of $L(K)$. Hence, we possess all the elements needed to construct λ_1^{inf}: it suffices to implement a strategy that plays as λ_1^{cmb}, but sequentially increasing the values of K and $L(K)$ up to infinity. Formally, let $(K_i)_{i \in \mathbb{N}}$ be a strictly increasing sequence of naturals, and for all $i \geq 0$, let $L(K_i)$ be the natural given by Def. 12.21. The strategy λ_1^{inf} is defined as follows:

- Initialize i to 0.

(a) Play λ_1^e for K_i steps and memorize $\mathsf{Sum} \in \mathbb{Z}$, the sum of weights encountered during these K_i steps.

(b) If $\mathsf{Sum} > 0$, then go to *(a)*.

Else, play λ_1^{wc} during $L(K_i)$ steps, then increment i and go to *(a)*.

By doing so, it is possible to show that λ_1^{inf} ensures an expected mean-payoff exactly equal to ν^*, as well as the worst-case requirement. For the worst-case, it suffices to apply the reasoning developed in Lemma 12.23. To show that λ_1^{inf} achieves the expected value ν^*, the intuitive argument is that the probability that a period of type *(a)* is followed by a period of type *(b)* tends to zero as K_i grows, since $\nu^* > 0$. Therefore, the probability that λ_1^{wc} is played infinitely many times is zero.

To illustrate this point, let us consider the example of Fig. 12.1. By playing λ_1^{inf} as defined above, $\mathcal{P}_1$ can ensure the worst-case requirement and induce the optimal expected mean-payoff 2, because the proportion of time spent following strategy λ_1^e will tend to one as the parameter K_i tends to infinity.

CHAPTER 13

Beyond Worst-Case Shortest Path

Introduction ◇ Pseudo-Polynomial-Time Algorithm ◇ Memory Requirements ◇ NP-Hardness of the Decision Problem

We study the BWC framework for the shortest path problem. That is, a generalization of the classical graph problem where $\mathcal{P}_1$ wants to ensure reachability of a target set while minimizing cost-to-target.

In Sect. 13.2, we establish a pseudo-polynomial-time algorithm to solve the BWC problem. Synthesized strategies require at most pseudo-polynomial memory, and we prove that such memory is necessary in general (Sect. 13.3).

In contrast to the mean-payoff case, our algorithm shows a complexity leap with regard to the individual worst-case and expected value threshold problems. We establish that the BWC problem is inherently harder as we prove its NP-hardness in Sect. 13.4. Hence this problem cannot be solved in truly-polynomial time unless P = NP.

Those results are joint work with Bruyère, Filiot and Raskin [BFRR13, BFRR14a, BFRR14b].

13.1 Introduction

Let us consider a game $G = (\mathcal{G}, S_1, S_2)$ with an underlying graph $\mathcal{G} = (S, E, w)$ such that the weight function $w \colon E \to \mathbb{N}_0$ assigns *strictly positive* integer weights to all edges, and a target set of states $T \subseteq S$ that $\mathcal{P}_1$ wants to reach with a path of bounded value. That is, $\mathcal{P}_1$ aims to ensure some threshold on the *truncated sum* value function TS_T.

In other words, we study the BWC synthesis problem for the *shortest path problem* (Sect. 2.3.2). More precisely, given an initial state $s_{\mathsf{init}} \in S$, the goal of $\mathcal{P}_1$ is to ensure to reach T with a path of (truncated) sum strictly lower than the threshold $\mu \in \mathbb{N}$ (we assume a natural threshold w.l.o.g. as all weights take positive integer values and so does the truncated sum function for any play reaching T) against all possible behaviors of $\mathcal{P}_2$ while guaranteeing, at the same time, an expected cost to target strictly lower than the threshold $\nu \in \mathbb{Q}$ against the finite-memory stochastic model of the adversary specified by the SOMM $\mathcal{M}(\lambda_2^{\mathsf{stoch}})$. Notice that regarding Def. 11.3, the inequalities are reversed. Hence we assume that $\nu < \mu$. Equivalently, the problem could be stated with value function $-\mathsf{TS}_T$ without changing the definition.

13.2 Pseudo-Polynomial-Time Algorithm

To solve the decision problem, we proceed as follows. First, we show how to construct, from the original game G and the worst-case threshold μ, a new game G_μ such that there is a one-to-one correspondence between the strategies of $\mathcal{P}_1$ in G_μ and the strategies of $\mathcal{P}_1$ in the original game G that are winning for the worst-case requirement (eq. (11.1)).

To construct this game, we unfold the original graph $\mathcal{G}$, tracking the current value of the truncated sum *up to the worst-case threshold* μ, and integrating this value in the states of an expanded graph $\mathcal{G}'$. In the corresponding game G', we then compute the set of states R from which $\mathcal{P}_1$ can reach the target set with cost lower than the worst-case threshold and we define the subgame $G_\mu = G' \restriction R$ such that any path in the graph of G_μ satisfies the worst-case requirement.

Second, from this new game G_μ and the SOMM $\mathcal{M}(\lambda_2^{\mathsf{stoch}})$ representing the stochastic model of the adversary, we construct an MDP in which we search

for a $\mathcal{P}_1$ strategy that ensures reachability of T with an expected cost strictly lower than ν (eq. (11.2)). If such a strategy exists, it is guaranteed that it will also satisfy the worst-case requirement against any strategy of $\mathcal{P}_2$ thanks to the bijection evoked earlier.

Remark 13.1. Observe that this algorithm is conceptually much simpler than what we presented in Chap. 12 for the mean-payoff value function. In particular, it follows a sequential approach: first it deals with the worst-case requirement, then it optimizes the expected value among the strategies that are safe with regard to the worst-case.

There is a key difference between mean-payoff and shortest path games that permits this sequential approach. In shortest path games, the set of *all* worst-case winning strategies can be represented as a finite game, based on the unfolding of the original game up to the worst-case bound. In the mean-payoff setting, there is no obvious finite representation of the winning strategies. Hence, it is not possible to follow a sequential approach without losing completeness of the algorithm. ◁

This sequential algorithm is depicted through the following example.

Example 13.2. Consider the game G depicted in Fig. 13.1.[1] We want to synthesize a BWC strategy of $\mathcal{P}_1$ that minimizes the expected truncated sum up to the target set $\{s_3\}$ under the worst-case threshold $\mu = 8$.

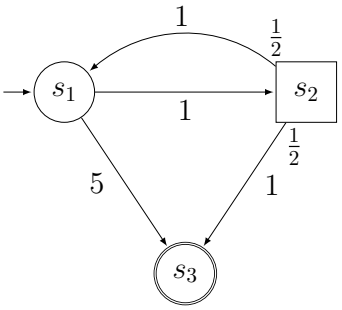


Figure 13.1: Simple BWC shortest path game with target set $\{s_3\}$ and worst-case threshold $\mu = 8$.

[1] We allow a deadlock on the target state for simplicity: it does not change the problem by definition of the truncated sum value function.

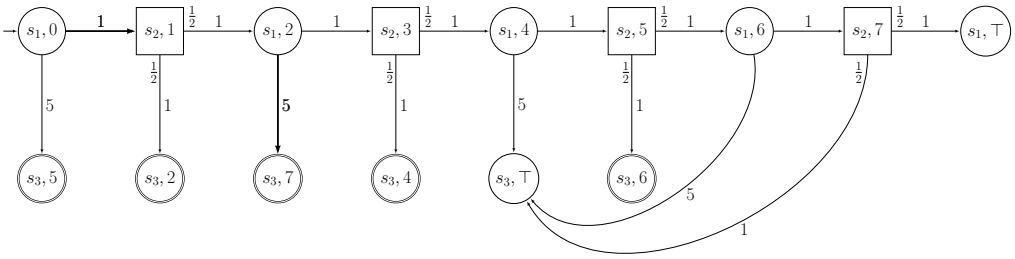


Figure 13.2: Unfolding of the game of Fig. 13.1: worst-case winning requires to reach a double state. Thick edges represent the strategy that minimizes the expected cost while ensuring this worst-case.

First, we unfold this game G up to the worst-case threshold (excluded), and obtain the game G' represented in Fig. 13.2. Observe that as soon as the worst-case threshold is reached, we stop the unfolding and associate symbol $\top$: the worst-case requirement is lost if such states are reached. This guarantees a finite (and at most pseudo-polynomial size) unfolding.

Therefore, it is clear that a BWC strategy of $\mathcal{P}_1$ must ensure reachability of states of G' that represent reaching the target state with a truncated sum strictly less than the worst-case threshold. Those states are depicted by double circles in the figure. Hence, $\mathcal{P}_1$ must stay within the attractor of those double states. It implies that state $(s_2, 3)$ of the unfolding and subsequent states are off-limits.

Knowing that, it now suffices to minimize the expected value within the safe region, which is achieved by the memoryless (with regard to G') strategy that chooses to go in $(s_2, 1)$ from $(s_1, 0)$ and to $(s_3, 7)$ from $(s_1, 2)$. This strategy is depicted by the thick edges on the figure. Observe that this strategy is memoryless in G', hence requires at most pseudo-polynomial memory in G. ◁

Theorem 13.3. *The beyond worst-case problem for the shortest path can be solved in pseudo-polynomial time: polynomial in the size of the underlying game graph, the SOMM for the stochastic model of the adversary and the encoding of the expected value threshold, and polynomial in the value of the worst-case threshold.*

Proof. Let $G = (\mathcal{G}, S_1, S_2)$ be the two-player game, $\mathcal{G} = (S, E, w)$ its underlying

graph, $w\colon E \to \mathbb{N}_0$ its weight function, $s_{\mathsf{init}} \in S$ the initial state, $T \subseteq S$ the target set, $\lambda_2^{\mathsf{stoch}} \in \Lambda_2^F$ the stochastic model of $\mathcal{P}_2$, with $\mathcal{M}(\lambda_2^{\mathsf{stoch}}) = (\mathsf{Mem}, \mathsf{m}_0, \alpha_{\mathsf{u}}, \alpha_{\mathsf{n}})$ its SOMM, $\mu \in \mathbb{N}$ the worst-case threshold, and $\nu \in \mathbb{Q}$ the expected value threshold.

Based on G and μ, we define the game $G' = (\mathcal{G}', S'_1, S'_2)$. Its underlying graph $\mathcal{G}' = (S', E', w')$ is built by unfolding the original graph $\mathcal{G}$, tracking the current value of the truncated sum *up to the worst-case threshold* μ, and integrating this value in the states of $\mathcal{G}'$. Formally, we have that

- $S'_1 = S_1 \times (\{0, 1, \ldots, \mu - 1\} \cup \{\top\})$, $S'_2 = S_2 \times (\{0, 1, \ldots, \mu - 1\} \cup \{\top\})$, and $S' = S'_1 \cup S'_2$;
- $E' = \big\{\big((s_1, u_1), (s_2, u_2)\big) \in S' \times S' \mid (s_1, s_2) \in E \wedge u_2 = u_1 + w((s_1, s_2))\big\}$, with the convention that, for all $c \in \mathbb{N}$, $\top + c = \top$, and, for all $u \in \mathbb{N}$, $u + c = \top$ if $u + c \geq \mu$;
- $\forall\, e = \big((s_1, u_1), (s_2, u_2)\big) \in E'$, $w'(e) = w((s_1, s_2))$.

The symbol $\top$ represents costs exceeding the worst-case threshold μ. The initial state in G' is $s'_{\mathsf{init}} = (s_{\mathsf{init}}, 0)$, and the target set is $T' = T \times \{0, 1, \ldots, \mu - 1\}$, i.e., in G' the target set is restricted to copies of states of the original target set that are reached with a sum less than μ. Notice that for any state $s'_1 = (s_1, \top) \in S'$, all its successors in $\mathcal{G}'$ are of the form $s'_2 = (s_2, \top)$.

Now, we compute in G' the set of states $R \subseteq S'$ from which $\mathcal{P}_1$ has a strategy to force reaching T' using a classical attractor computation, i.e., $R = \mathsf{Attr}^{\mathcal{P}_1}_{G'}(T')$. Clearly, all states outside this attractor set are losing for the worst-case requirement. Indeed, from any state outside of R, either $\mathcal{P}_1$ cannot force reaching a state $s' = (s, u)$ with $s \in T$, or he can only do it for $u = \top$. In particular, if $s'_{\mathsf{init}} = (s_{\mathsf{init}}, 0) \notin R$ then we know that $\mathcal{P}_1$ cannot enforce the worst-case threshold in the original game G, and we can stop here in this case: no strategy exists for the BWC problem.

Assume that $s'_{\mathsf{init}} = (s_{\mathsf{init}}, 0) \in R$. Let us write $G_\mu = G' \restriction R$. Note that there will be deadlocks in G_μ (i.e., states with no successors): this is guaranteed since the sum of weights is strictly growing (recall $w\colon E \to \mathbb{N}_0$) and states of the form $(s, \top)$ do not belong to R by definition. However, the only deadlocks will be on states that are in $T' \subseteq R$ (by definition of R as the attractor of T'). Hence, we easily get rid of them by adding self-loops of weight zero (this does not change

the truncated sum of paths up to T' by definition of $\mathsf{TS}_{T'}$). It is easy to see that all strategies $\lambda_1 \in \Lambda_1(G_\mu)$ are winning for the worst-case requirement, and that there is a bijection between the winning strategies for the worst-case requirement in the original game G and the strategies in G_μ.

We are now equipped to handle the expectation objective. We proceed as follows. First, we take the product of G_μ and $\mathcal{M}(\lambda_2^{\mathsf{stoch}}) = (\mathsf{Mem}, \mathsf{m}_0, \alpha_\mathsf{u}, \alpha_\mathsf{n})$, following the construction of Lemma 12.5 (the proof holds for the truncated sum value function as well). On the product game, we again preserve correspondence with the worst-case winning strategies in the original game G. Applying the memoryless stochastic model (resulting of Lemma 12.5) on the product game, we obtain an MDP P. It is then clear that $\mathcal{P}_1$ has a strategy to enforce an expected value strictly less than the threshold ν in P *if and only if* $\mathcal{P}_1$ has a strategy that *both* enforces the worst-case threshold against any strategy $\lambda_2 \in \Lambda_2(G)$, and the expectation threshold against $\lambda_2^{\mathsf{stoch}}$ in G. To decide if such a strategy exists, we compute the minimal achievable expected value on P and we compare it against the threshold ν. Thanks to the reduction from truncated sum to total-payoff proposed in Sect. 2.3.2, we know that this optimal value can be achieved by a memoryless strategy and its computation can be executed in polynomial time in the size of the encoding of P via linear programming. Hence, it requires time polynomial in the size of the encoding of G and $\mathcal{M}(\lambda_2^{\mathsf{stoch}})$, and polynomial in the value μ (since $|S'| = |S| \cdot (\mu + 1)$). □

13.3 Memory Requirements

In Thm. 13.4, we characterize the memory needed by strategies satisfying the BWC problem. The construction developed in Thm. 13.3 yields an upped bound for the memory that is polynomial in the size of the game and the stochastic model, and in the value of the worst-case threshold. Indeed, the synthesized strategy is memoryless in the MDP P that is obtained by taking the product of the expanded game G_μ, such that $|G_\mu| \leq |G| \cdot (\mu+1)$, with the SOMM $\mathcal{M}(\lambda_2^{\mathsf{stoch}})$. Hence the memory needed is bounded by a polynomial in the sizes $|G|$ and $|\mathcal{M}(\lambda_2^{\mathsf{stoch}})|$, and in the value μ.

We also exhibit a family of games (Fig. 13.3) for which winning the BWC problem requires memory linear in μ, hence proving that the pseudo-polynomial

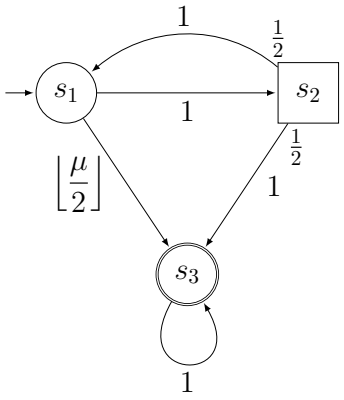


Figure 13.3: Family of games requiring memory linear in μ for the BWC problem.

bound is tight. The intuition is as follows. Assume a worst-case threshold $\mu \in \{13 + k \cdot 4 \mid k \in \mathbb{N}\}$ (the set is defined in order to ease computations). From state s_1, $\mathcal{P}_1$ can ensure reaching the target set $T = \{s_3\}$ at a guaranteed cost of $\lfloor \frac{\mu}{2} \rfloor$. Nevertheless, in order to *minimize* the expected cost of reaching T, $\mathcal{P}_1$ should try to reach it via state s_2, as the cost will be diminished. Hence, $\mathcal{P}_1$ should play edge (s_1, s_2) repeatedly, up to the point where playing (s_1, s_3) becomes mandatory to preserve the worst-case requirement (i.e., when the running sum of weights becomes equal to $\lfloor \frac{\mu}{2} \rfloor$ as the total cost for the worst outcome will be $2 \cdot \lfloor \frac{\mu}{2} \rfloor < \mu$). To implement this strategy (Fig. 13.4), $\mathcal{P}_1$ has to play (s_1, s_2) exactly $\lfloor \frac{\mu}{4} \rfloor$ times and then switch to (s_1, s_3). Clearly, this requires memory linear in μ. The expected value threshold ν can be chosen sufficiently low so that $\mathcal{P}_1$ is compelled to use this optimal strategy to satisfy the BWC problem.

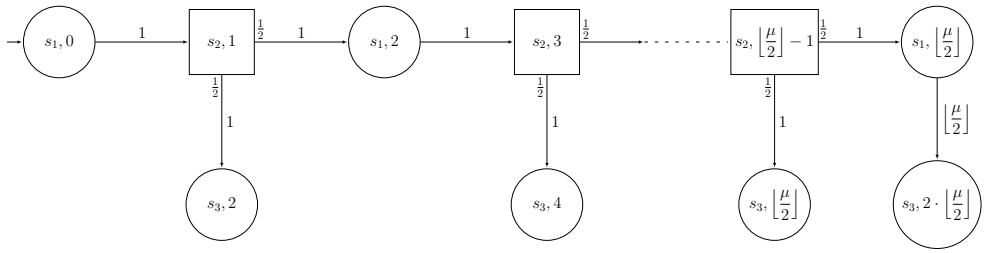


Figure 13.4: Partial representation of the MC induced by the BWC strategy that minimizes the expected cost to target in $G(\mu)$, $\mu \in \{13 + k \cdot 4 \mid k \in \mathbb{N}\}$.

Theorem 13.4. *Memory of pseudo-polynomial size may be necessary and is always sufficient to satisfy the BWC problem for the shortest path: polynomial in the size of the game and the stochastic model, and polynomial in the worst-case threshold value.*

Proof. The upper bound on the size of the memory can be obtained directly from the construction exposed in the proof of Thm. 13.3. Indeed, we have shown that if the BWC problem can be satisfied, the memoryless strategy that minimizes the expectation in the MDP P does satisfy it. Translated back to the original game, this strategy has a memory which is polynomial in $|G|$, $|\mathcal{M}(\lambda_2^{\mathsf{stoch}})|$, and the value of μ. Intuitively, the strategy needs to memorize the current value of the sum of weights, up to the value of the worst-case threshold (at which point it does not matter to bookkeep it anymore as $\mathcal{P}_1$ has already failed to enforce the worst-case requirement). Hence, such a strategy requires memory polynomial in the input game and the stochastic model, and in the threshold.

To prove that pseudo-polynomial memory may be necessary, we introduce a family of games $(G(\mu))_{\mu\in\{13+k\cdot 4 | k\in\mathbb{N}\}}$, indexed by the value of the worst-case threshold. This value is taken in a specific set $\{13 + k \cdot 4 \mid k \in \mathbb{N}\}$ mostly to ease the following calculations. The family is presented in Fig. 13.3: it consists of three states $S = \{s_1, s_2, s_3\}$. The weight function only assigns strictly positive weights as assumed in the setting of the shortest path problem. All weights are equal to 1 except for edge (s_1, s_3) which has a weight $\left\lfloor \frac{\mu}{2} \right\rfloor$. Notice that μ is chosen odd and such that $\left\lfloor \frac{\mu}{2} \right\rfloor$ is even.

We will consider the values of the expectation threshold ν that can be ensured by a BWC strategy in such a game, under the chosen worst-case threshold μ and against a stochastic model assigning uniform distributions, and show that to minimize this value, $\mathcal{P}_1$ needs to use linear memory in μ, hence proving the claim.

First, observe that if the running sum of weights (which is an integer value) gets strictly larger than $\left\lfloor \frac{\mu}{2} \right\rfloor$, then $\mathcal{P}_1$ loses the worst-case requirement (eq. (11.1)) as playing (s_1, s_2) does not guarantee reaching T, and playing (s_1, s_3) induces a total cost at least equal to μ. Hence, when in s_1 with a running sum equal to $\left\lfloor \frac{\mu}{2} \right\rfloor$, $\mathcal{P}_1$ has no valid choice but to take the edge (s_1, s_3). Since randomization clearly does not help (as it will produce consistent outcomes that are losing if the edge (s_1, s_2) is repeatedly assigned a non-zero probability), defining the optimal

strategy of $\mathcal{P}_1$ boils down to deciding for how long he should take the edge (s_1, s_2) before switching (if at all).

We claim that it should maximize the number of passes in s_2 (Fig. 13.4). Let n denote the number of times $\mathcal{P}_1$ chooses (s_1, s_2) before switching. Clearly, to guarantee satisfaction of the worst-case requirement, we need $2 \cdot n + \left\lfloor \frac{\mu}{2} \right\rfloor < \mu$. Since the threshold is odd, we have $2 \cdot n + \left\lfloor \frac{\mu}{2} \right\rfloor < 2 \cdot n + \frac{\mu}{2}$. Hence, it suffices to have $2 \cdot n + \frac{\mu}{2} \leq \mu$, or equivalently, $n \leq \frac{\mu}{4}$. Note that this bound is linear in the value of μ.

What remains to prove is that increasing the number of passes results in a decrease of the expected value. Let $e(n)$ denotes the expected value induced by the strategy that plays edge (s_1, s_2) for n times before switching. Careful computation reveals that $e(n)$ can be expressed as follows:

$$e(n) = \sum_{i=0}^{n-1} \frac{1}{2^{i-1}} + \frac{1}{2^n} \cdot \left\lfloor \frac{\mu}{2} \right\rfloor. \tag{13.1}$$

Our thesis is that for all $n \geq 0$, $e(n) < e(n-1)$. By eq. (13.1), that is

$$\begin{gathered}
\sum_{i=0}^{n-1} \frac{1}{2^{i-1}} + \frac{1}{2^n} \cdot \left\lfloor \frac{\mu}{2} \right\rfloor < \sum_{i=0}^{n-2} \frac{1}{2^{i-1}} + \frac{1}{2^{n-1}} \cdot \left\lfloor \frac{\mu}{2} \right\rfloor, \\
\frac{1}{2^{n-2}} - \frac{1}{2^n} \cdot \left\lfloor \frac{\mu}{2} \right\rfloor < 0, \\
\left\lfloor \frac{\mu}{2} \right\rfloor > \frac{2^n}{2^{n-2}} = 4, \\
\mu > 9,
\end{gathered}$$

and the last inequality is granted thanks to the hypothesis that $\mu \in \{13 + k \cdot 4 \mid k \in \mathbb{N}\}$. This shows that increasing n decreases the expectation, as wanted.

In conclusion, the optimal BWC strategy for the expected value criterion consists in playing (s_1, s_2) for exactly $n = \left\lfloor \frac{\mu}{4} \right\rfloor$ times, then swithing to (s_1, s_3) to ensure the worst-case (the corresponding MC is represented in Fig. 13.4). Following our computations, it is possible to impose that playing this strategy is necessary to satisfy the BWC problem by taking the expected value threshold such that $e(n) < \nu \leq e(n-1)$. This proves that memory linear in μ is needed for the given family of games. □

Remark 13.5. In contrast to the case of the mean-payoff value function (as presented in Sect. 12.7), infinite memory gives no additional power in the shortest path context. Indeed, the proof of Thm. 13.3 gives a complete representation of worst-case winning strategies through the game G_μ and it is further proved that finite memory suffices to define an optimal strategy with regard to the expected value criterion among these worst-case winning strategies. ◁

13.4 NP-Hardness of the Decision Problem

We conclude our study of the BWC problem in the shortest path setting (i.e., for the truncated sum value function) by showing that it is very unlikely that a truly-polynomial (i.e., also polynomial in the size of the encoding of the worst-case threshold) time algorithm exists, as we establish in Thm. 13.6 that the decision problem is NP-hard. Actually, it is likely that the problem is not in NP at all, since we prove a reduction from the K^{th} *largest subset problem* which is known to be NP-hard and commonly thought to be outside NP as natural certificates for the problem are larger than polynomial [GJ79].

The K^{th} largest subset problem is expressed as follows. Given a finite set A, a size function $h\colon A \to \mathbb{N}_0$ assigning strictly positive integer values to elements of A, and two naturals $K, L \in \mathbb{N}$, decide if there exist K distinct subsets $C_i \subseteq A$, $1 \leq i \leq K$, such that $h(C_i) = \sum_{a \in C_i} h(a) \leq L$ for all K subsets. The NP-hardness of this problem was proved in [JK78] via a Turing reduction from the partition problem.

The key steps of the reduction are as follows. We build a game composed of two gadgets. The *random subset selection gadget* (Fig. 13.5) stochastically generates paths that represent subsets of A. It has the important property that all subsets are equiprobable.

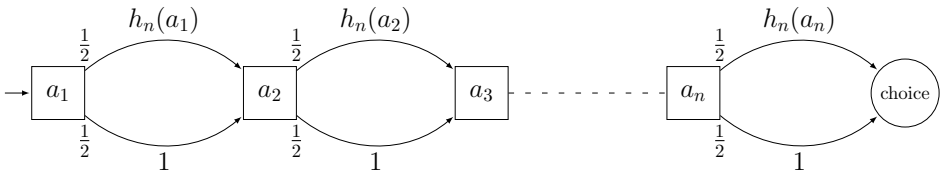


Figure 13.5: Random subset selection gadget: an element is selected in the subset if the upper edge is taken when leaving the corresponding state.

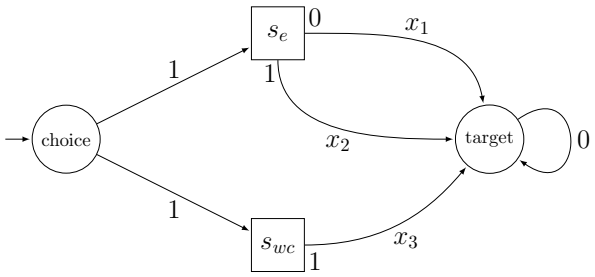


Figure 13.6: Choice gadget: choosing s_e is best for the expected value, but it is safe with regard to the worst-case if and only if the random subset selection produced a subset C such that $h(C) \leq L$.

The *choice gadget* follows (Fig. 13.6). In it, $\mathcal{P}_1$ decides either to go to s_e, which leads to lower expected values (and lower is better in our setting) but may be dangerous for the worst-case requirement, or to go to s_{wc}, which is always safe with regard to the worst-case threshold but induces an higher expected cost. The trick is to prove that we can define values of the thresholds and the weights used in the gadgets such that an optimal[2] strategy for $\mathcal{P}_1$ consists in choosing state s_e only when the randomly generated subset $C \subseteq A$ satisfies $h(C) \leq L$, as asked by the K^{th} largest subset problem; and such that this strategy satisfies the BWC problem if and only if there exist K distinct subsets that verify this bound, i.e., if and only if the answer to the K^{th} largest subset problem is Yes.

Theorem 13.6. *The beyond worst-case problem for the shortest path is NP-hard.*

Proof. We establish a reduction from the K^{th} largest subset problem: given a finite set $A = \{a_1, \ldots, a_n\}$ (hence $n = |A|$), a size function $h\colon A \to \mathbb{N}_0$, and two naturals $K, L \in \mathbb{N}$, decide if there exist K distinct subsets $C_i \subseteq A$, $1 \leq i \leq K$, such that $h(C_i) = \sum_{a \in C_i} h(a) \leq L$ for all K subsets. This problem is known to be NP-hard [JK78, GJ79]. Note that the restriction to $\mathbb{N}_0$ for the codomain of h in place of $\mathbb{N}$ is w.l.o.g. as the problem is satisfied for A, K, L and $h\colon A \to \mathbb{N}$ if and only if it is satisfied for $A' = A \setminus \{a \in A \mid h(a) = 0\}$, $K' = \left\lfloor \frac{K}{2^{|A|-|A'|}} \right\rfloor$, $L' = L$ and $h'\colon A' \to \mathbb{N}_0$ such that for all $a \in A'$, $h'(a) = h(a)$. Obviously, we should have $K \leq 2^n$, otherwise the problem is trivial since we cannot find a sufficient number of *distinct* subsets.

[2] Minimizing the expectation while guaranteeing a given worst-case threshold.

Before giving the details of our reduction, we define, given A and h, the function $h_n \colon A \to \mathbb{N}_0$ such that for each $a \in A$, $h_n(a) = (n+1) \cdot h(a)$. Clearly, it satisfies the following property:

$$\forall C \subseteq A,\ h(C) \leq L \Leftrightarrow h_n(C) \leq (n+1) \cdot L. \tag{13.2}$$

We now present two gadgets useful to construct a game and an associated BWC shortest path problem such that the answer to the K^{th} largest subset problem is YES if and only if the answer to the BWC problem is also YES.

First, the fragment of the game graph depicted in Fig. 13.5 is called the *random subset selection gadget*. All its states belong to $\mathcal{P}_2$, except for the last one, and model the selection (or not) of an element of A in a subset. Basically, there is a bijection between paths[3] in this gadget and subsets of A: an element $a_i \in A$ is selected by the gadget if the outgoing upper edge is taken when leaving state a_i, and not selected when the outgoing lower edge is taken. The stochastic model followed by $\mathcal{P}_2$ in the BWC shortest path problem we construct is the uniform distribution: the upper and lower edges are equiprobable in all states. This gadget verifies the following important properties.

1. All subsets are equiprobable: they have probability $\frac{1}{2^n}$ to be selected.
2. If the gadget selects a subset $C \subseteq A$ through the corresponding path p_C, the total sum of weights along p_C, denoted by $t(p_C)$, is equal to $h_n(C)+n-|C|$.

By eq. (13.2), we have that

$$\forall C \subseteq A,\ h(C) = L \Leftrightarrow (n+1) \cdot L \leq t(p_C) < (n+1) \cdot (L+1). \tag{13.3}$$

Indeed, consider the following. Observe that $0 \leq n - |C| \leq n$, for any subset $C \subseteq A$. Hence the left-to-right implication is trivial. For the converse, we directly deduce the following equivalent expression:

$$L - \frac{n-|C|}{n+1} \leq h(C) < L + 1 - \frac{n-|C|}{n+1}.$$

[3] To be able to formally distinguish between such paths, which we usually define as sequence of *states*, we should introduce dummy states to split edges. We omit this technical trick for the sake of simplicity.

The left inequality implies that $L-1 < h(C)$, and since $h(C) \in \mathbb{N}$, that $L \leq h(C)$. The right inequality implies that $h(C) < L+1$, and using the same argument, that $h(C) \leq L$. We conclude that eq. (13.3) is true. Consequently, we define the value $T = (n+1) \cdot (L+1) - 1$, which is an upper bound on the value $t(p_C) \leq T$ of a path corresponding to a subset $C \subseteq A$ such that $h(C) \leq L$.

Now consider the second gadget, called the *choice gadget* and depicted in Fig. 13.6. This gadget comes after the random subset selection gadget. Its aim is to discriminate subsets generated by the preceding gadget based on whether or not they satisfy the upper bound $h(C) \leq L$. Observe the shared *choice* state. There, $\mathcal{P}_1$ has the choice to go up to state s_e or down to state s_{wc}. Both belong to $\mathcal{P}_2$. Again, probabilities for the stochastic model of the adversary are depicted in Fig. 13.6. So, in s_e, an arbitrary strategy of $\mathcal{P}_2$ can decide to impose cost x_1 or cost x_2 before reaching the target set of the game (notice we have set the weight of the self-loop to zero on the target set, as discussed previously). Nonetheless, the stochastic model $\lambda_2^{\mathsf{stoch}}$ of $\mathcal{P}_2$ assigns probability zero to the edge of weight x_1: the expectation of any strategy of $\mathcal{P}_1$ against this stochastic model will be independent of the value x_1. In s_{wc}, the cost added is always equal to x_3. Intuitively, we will choose values so that to minimize his expected cost-to-target, $\mathcal{P}_1$ should choose s_e, but also so that the worst-case requirement implies that it is only safe to choose this state if the previous path defined a subset that satisfies the bound $h(C) \leq L$ given by the K^{th} largest subset problem.

To complete the description of the reduction, we need to precise the values of the thresholds μ and ν, and the weights x_1, x_2 and x_3. Assume that we choose the worst-case threshold and the weights such that:

(a) $T+1+x_1+1 \geq \mu$, i.e., going to s_e with a path p_C (obtained in the random subset selection gadget) of cost $t(p_C) > T$ (i.e., with a selected subset $C \subseteq A$ such that $h(C) > L$ by eq. (13.3)) is losing for the worst-case threshold if $\mathcal{P}_2$ takes the edge of weight x_1;

(b) $T + x_1 + 1 < \mu$ and $T + x_2 + 1 < \mu$, i.e., going to s_e with a path p_C of cost $t(p_C) \leq T$ (i.e., $h(C) \leq L$) is safe for the worst-case requirement whatever the choice of $\mathcal{P}_2$;

(c) for all $C \subseteq A$, we have that $t(p_C) + x_3 + 1 < \mu$, i.e., going to s_{wc} is always safe for the worst-case requirement.

Then clearly, $\mathcal{P}_1$ can always choose to go to s_{wc} and ensure the worst-case threshold, but he can go up only if the chosen subset C satisfies $h(C) \leq L$, which is equivalent to say that $t(p_C) \leq T$. We add the following constraints to the choices of the expectation threshold and the weights:

(d) in order to minimize the expected truncated sum in the MDP defined by the stochatic model, the optimal choice for $\mathcal{P}_1$ is to always take s_e when possible (i.e., when $h(C) \leq L$, or equivalently $t(p_C) \leq T$ because of the constraint *(a)* defined above);

(e) the expected value ν^* of this optimal choice satisfies the expectation requirement (i.e., $\nu^* < \nu$) if and only if the number of distinct subsets $C_i \subseteq A$ verifying $h(C_i) \leq L$ is larger than or equal to K.

We will now define values such that properties *(a)* through *(e)* are ensured. First, let $Q = \max\{t(p_C) \mid C \subseteq A\} = t(p_A)$ be the maximal cost of a path in the random subset selection gadget (the equality with $t(p_A)$ is thanks to the size function h assigning strictly positive values). We claim the needed properties are verified for the following values:

$$\mu = 2^{n+1} \cdot n \cdot (Q+2), \qquad \nu = \frac{K \cdot (T+2) + (2^n - K) \cdot \mu}{2^n},$$

$$x_1 = \mu - T - 2, \qquad x_2 = 1, \qquad x_3 = \mu - Q - 2.$$

Using these, we review each property one-by-one. For *(a)*, we obtain by simple substitutions

$$\textit{(a)} \quad \Leftrightarrow \quad T + 1 + \mu - T - 2 + 1 \geq \mu \quad \Leftrightarrow \quad 0 \geq 0,$$

which is obviously true. Similarly, for *(b)*, we have that

$$\begin{aligned} \textit{(b)} \quad &\Leftrightarrow \quad (T + \mu - T - 2 + 1 < \mu) \wedge (T + 1 + 1 < \mu) \\ &\Leftrightarrow \quad (-1 < 0) \wedge (T + 2 < 2^{n+1} \cdot n \cdot (Q+2)). \end{aligned} \tag{13.4}$$

The first term of the conjunction is trivially true so we focus on the second one. Without loss of generality, we can assume that $L < h(A)$ as otherwise the

K^{th} largest subset problem reduces to decide if $K \leq 2^n$. Thus, we deduce the inequality $T < (n+1) \cdot (h(A)+1) - 1$. Also note that, by definition, we have that $Q = t(p_A) = h_n(A) = (n+1) \cdot h(A)$. Using these inequalities in eq. (13.4), we derive that proving the following central inequality suffices to obtain *(b)*:

$$T + 2 < (n+1) \cdot (h(A)+1) + 1 \leq 2^{n+1} \cdot n \cdot ((n+1) \cdot h(A) + 2) = 2^{n+1} \cdot n \cdot (Q+2).$$

This boils down to

$$(2^{n+1} \cdot n - 1) \cdot (n+1) \cdot h(A) + (2 \cdot 2^{n+1} - 1) \cdot n - 2 \geq 0,$$

which is true for $n \geq 1$ (which we can assume otherwise $A = \emptyset$ and the problem is trivial). Hence, property *(b)* is verified by our choice of values. Now, consider property *(c)*. We have

$$(c) \quad \Leftrightarrow \quad t(p_C) + \mu - Q - 2 + 1 < \mu \quad \Leftrightarrow \quad t(p_C) < Q + 1,$$

which is true by definition of Q as the maximum over the values of paths. Regarding property *(d)*, we have to show that choosing s_e gives an expectation strictly lower (recall we want to minimize it) than choosing s_{wc}. Observe that due to the particular structure of the game graph, the strategy of $\mathcal{P}_1$ is restricted to this one-shot choice of edge. Note that in this expected value context, the actual value obtained in the random subset selection gadget does not matter to decide whether to go to s_e or to s_{wc}: hence it suffices to look at the expectation from the *choice* state up to the *target* state. For s_e, it is trivially equal to $1 + 1 = 2$ as the stochastic model $\lambda_2^{\mathsf{stoch}}$ of $\mathcal{P}_2$ always chooses the edge of weight x_2. For s_{wc}, this expectation is equal to

$$\begin{aligned} 1 + x_3 = 1 + \mu - Q - 2 &= 2^{n+1} \cdot n \cdot (Q+2) - Q - 1 \\ &\geq (2^{n+1} \cdot n - 1) \cdot (Q+2) \geq (Q+2) > 2, \end{aligned}$$

and we obtain the claim *(d)*. Note that an actual strategy that satisfies the BWC problem will only be able to choose s_e if the selected path satisfies the bound $t(p_C) \leq T$, as discussed in properties *(a)* and *(b)*.

Finally, it remains to show the most involved property *(e)*: proving it will conclude our reduction as we will obtain that the answer to the K^{th} largest subset

problem is YES if and only if the answer to the BWC problem we have defined is YES. Note that combining the already proved properties *(a)* through *(d)*, we know that the strategy $\lambda_1 \in \Lambda_1^{PF}$ that chooses state s_e when $t(p_C) \leq T$ and state s_{wc} otherwise, yields the minimal expectation value ν^* *under the worst-case constraint* of threshold μ. Hence, it suffices to study this strategy to answer the BWC problem. Our claim is thus that

$$\nu^* = \mathbb{E}_{a_1}^{G[\lambda_1, \lambda_2^{\mathsf{stoch}}]} < \nu \quad \Leftrightarrow \quad \Big| \{C \subseteq A \mid h(C) \leq L\} \Big| \geq K, \tag{13.5}$$

with G and $\lambda_2^{\mathsf{stoch}}$ the game and stochastic model we defined.

For the left-to-right implication, we reason by contradiction and show that if there is only $K-1$ (or less) distinct subsets whose sum is less than or equal to L, then strategy λ_1 has an expected cost larger than or equal to ν. To show that, we use the fact that all paths (i.e., subsets) have equal probability in the random subset selection gadget, and establish that a lower bound on the sum of all the paths under this strategy reaches or exceeds $2^n \cdot \nu$. Recall that $\mathcal{P}_2$ follows its stochastic model $\lambda_2^{\mathsf{stoch}}$ for this matter. First, let $\mathsf{LB}_e = 0$ which is trivially a lower bound for the cost of all the paths that goes through s_e. Second, let $\mathsf{LB}_{wc} = (2^n - (K-1)) \cdot (T+1+x_3+1)$: it is clearly a lower bound for the sum of the values of paths that go through state s_{wc} when $\mathcal{P}_1$ follows strategy λ_1. We have that $2^n \cdot \nu^* \geq \mathsf{LB}_e + \mathsf{LB}_{wc}$. Let us now establish that $\mathsf{LB}_{wc} \geq 2^n \cdot \nu$ and we will be done. We proceed as follows.

$$\begin{aligned}
\mathsf{LB}_{wc} - 2^n \cdot \nu &= (2^n - K + 1) \cdot (T + 1 + \mu - Q - 2 + 1) \\
&\qquad - K \cdot (T+2) - (2^n - K) \cdot \mu \\
&= \mu + (2^n - K + 1) \cdot (T - Q) - K \cdot (T+2) \\
&= 2^{n+1} \cdot n \cdot (Q+2) + (2^n - K + 1) \cdot (T - Q) - K \cdot (T+2)
\end{aligned}$$

Recall that $T, Q \geq 0$, $n \geq 1$ and $0 \leq K \leq 2^n$ (otherwise the answer is trivially NO). Furthermore, T is the upper bound on the values of paths p_C representing good subsets, i.e., subsets $C \subseteq A$ such that $h(C) \leq L$. This value is used by the strategy λ_1 implemented by $\mathcal{P}_1$ to decide whether going to s_e is safe with regard to the worst-case requirement or not. As such, we can assume that $T \leq Q$, otherwise all paths are safe and the answer to the problem is trivially YES (since

all subsets respect the bound and $K \leq 2^n$). Using $K \leq 2^n$, $T \geq 0$ and $T \leq Q$, we can write

$$\mathsf{LB}_{wc} - 2^n \cdot \nu \geq (2^{n+1} \cdot n - 2^n + K - 1 - K) \cdot Q + (2^{n+1} \cdot n \cdot 2 - 2 \cdot K). \quad (13.6)$$

To prove that this last expression is non-negative, we analyze its terms. We know that $Q \geq 0$. For its coefficient, we have

$$2^{n+1} \cdot n - 2^n - 1 \geq 2^n - 1 \geq 0$$

because $n \geq 1$. For the last term, we use $K \leq 2^n$ and obtain that

$$2^{n+2} \cdot n - 2 \cdot K \geq 2^{n+2} - 2^{n+1} = 2^{n+1} \geq 0.$$

Hence all terms of eq. (13.6) are non-negative and $\mathsf{LB}_{wc} \geq 2^n \cdot \nu$, proving that the left-to-right implication of eq. (13.5) is verified.

It remains to prove the right-to-left implication. Assume there are exactly K distinct subsets of sum less than or equal to L (if there are more, then the bounds below are easier to obtain). We claim that strategy λ_1 ensures an expected truncated sum ν^* strictly lower than ν. To show this, we establish that the total sum of the outcomes under this strategy of $\mathcal{P}_1$ and the stochastic model of $\mathcal{P}_2$ is strictly bounded from above by $2^n \cdot \nu$, and the claim follows thanks to all paths being equiprobable in the random subset selection gadget. First, consider the paths that go through s_e (i.e., all the paths corresponding to subsets C such that $h(C) \leq L$). By definition of λ_1 and our hypothesis, there are exactly K such paths. We define $\mathsf{UB}_e = K \cdot (T + 2)$, a clear upper bound for the sum of the values of these paths, by definition of T and $\lambda_2^{\mathsf{stoch}}$. Second, there are $(2^n - K)$ paths that go through s_{wc}. Let $\mathsf{UB}_{wc} = (2^n - K) \cdot (Q + 1 + x_3) = (2^n - K) \cdot (\mu - 1)$ be a bound for the sum of the values of all these paths. Clearly,

$$\begin{aligned} 2^n \cdot \nu^* \leq \mathsf{UB}_e + \mathsf{UB}_{wc} &= K \cdot (T + 2) + (2^n - K) \cdot (\mu - 1) \\ &< K \cdot (T + 2) + (2^n - K) \cdot \mu = 2^n \cdot \nu \end{aligned}$$

by definition of the expected value threshold ν, and so we are done for this direction.

Having verified both directions of the equivalence given in eq. (13.5), the

correctness of our reduction from the K^{th} largest subset problem is established. Note that it requires values of the thresholds that are exponential in the size of the set A and polynomial in the value of the largest weight assigned by the size function h (or equivalently, exponential in its encoding) for the K^{th} largest subset problem. It also requires to use edge weights that are polynomial in these values. Observe that this is not a problem, as all those values may be represented using a logarithmic number of bits, hence polynomially in the characteristics of the initial K^{th} largest subset problem. Finally, notice that we do need to consider exponential constants in our game to obtain the NP-hardness of our problem, as for values polynomial in the size of the game, the algorithm described in Sect. 13.2 actually operates in truly-polynomial time. This concludes our proof. □

Part V

Discussion

CHAPTER 14

Conclusion and Future Work

Conclusion ◇ Future Work

We first give a brief summary of our contributions in Sect. 14.1. We discuss our main models and results, and how they fit in the advocated shift toward *multi-criteria quantitative models*. We examine some limitations of our work and how they can be addressed. We also review precise questions left open by our thesis.

We close the thesis with an overview of promising research directions linked to our work, presented in Sect. 14.2. This last section is more speculative, mixing concrete questions under our study and long-term prospects.

14.1 Conclusion

Research focus. As discussed in Sect. 1.2, we started this thesis from an observation. Since the last decade, there has been a growing trend in the study of *quantitative models for verification and synthesis* (e.g., [CdAHS03, BCHJ09]). This is well-justified by the need to construct system controllers that are not only *functionally correct* but also *efficient*. To that end, quantitative extensions of the classical game-theoretic framework were introduced, resulting in many interesting models, both on the fundamental level and considering applicability (e.g., [EM79, BFL+08, GS09]). Some of the most well-known classes of games were discussed in Chap. 2.

A lot of different aspects can be considered in quantitative models. Techniques and results vary depending on the type of winning objective (e.g., mean-payoff, total-payoff, energy), on the underlying model (e.g., turn-based game, Markov decision process) or on the considered semantics for winning strategies (e.g., worst-case, expected value). However, most existing models can be characterized as *single-criterion*: it is not possible to express situations where several of these aspects are mixed.

While difficult questions remain open for single-criterion models, such as whether one-dimension mean-payoff games are in P or not [EM79, ZP96, Jur98, BCD+11], the research community has acknowledged the need for *multi-criteria* quantitative models. Indeed, in practice the performance of reactive systems is impacted by interplays and trade-offs between several criteria. Progress has been made in studying multi-dimension quantitative objectives [CDHR10, VR11], conjunctions with a parity objective [CHJ05, BMOU11, CD12], or even in the study of trade-offs between expected value and variance in stochastic models [MT11, BCFK13].

Our contributions participate in the *shift* from single-criterion quantitative models to multi-criteria quantitative models. We have studied three important settings: *multi-dimension quantitative games* (with mean-payoff, total-payoff, energy in conjunction with parity objectives), games with *window objectives* and the *beyond worst-case synthesis* framework (combining worst-case and expected value). Recall that a detailed overview of our most important results is presented in Chap. 3.

Multi-dimension objectives. In Chap. 4, we reviewed existing results for multi-dimension quantitative objectives and conjunction of a quantitative objective with a parity one. We presented the first study of *multi-dimension total-payoff games* and proved that they are undecidable for games with five dimensions or more. This is surprising as it demonstrates the loss of a strong similarity with the mean-payoff objective that is verified in the one-dimension case.

Open problem 1. *Can we prove (un)decidability of total-payoff games with* $k = 2, 3$ *or* 4 *dimensions?*

In Chap. 5, we gave *tight exponential bounds* on memory for multi-dimension energy objectives in conjunction with parity. Those results extend to multi-dimension mean-payoff parity games under finite-memory strategies. We improved existing results [BJK10] from a triple exponential to a single one. Bounding the complexity of controllers is a key concern in applications of synthesis.

We established an optimal synthesis algorithm for such strategies in Chap. 6. Our algorithm was conceived to be *symbolic* and *incremental* in order to be efficient in most applications despite the exponential worst-case bound. The practical relevance of our approach was proved by Bohy et al., who implemented an efficient synthesis tool for LTL specifications with mean-payoff objectives based on our algorithm [BBFR13].

Observe that our algorithm answers the *unknown initial credit problem*: it guarantees finding a winning strategy and a finite initial credit vector if they exist. However, it does not answer the *fixed initial credit problem* which is to decide, given a credit vector, if it suffices to win. This problem has been proved to be EXPSPACE-hard [BJK10, FJLS11] hence cannot be tackled by our algorithm unless EXPSPACE = EXPTIME. Abdulla et al. presented an algorithm that can solve this problem, partly based on our results, but its complexity is open [AMSS13]. The complexity gap between our EXPTIME algorithm and the EXPSPACE-hardness of the fixed initial credit problem indicates that there exist games on which our algorithm does not capture *all* incomparable credits. Still, we have not been able to generate such a game. This seemingly reveals that for most applications, the algorithm captures all winning credits, and that obtaining a witness game would require intricate technique.

Open problem 2. *Can we find a game for which our synthesis algorithm does not capture all winning credits?*

Finally, in Chap. 7, we gave a precise characterization of the possible trade-off of memory for randomness in games with multi-dimension mean-payoff, energy and parity objectives. We established that this trade-off is possible only in two-player one-dimension mean-payoff parity games and in one-player multi-dimension mean-payoff parity games. We argued that such strategies can be *conceptually simpler* to conceive, at the price of slightly relaxing the winning semantics.

Window objectives. In Chap. 8, we introduced the concept of *window objectives* and studied their relation with classical mean-payoff and total-payoff objectives. We believe that window objectives are interesting on the standpoint of *expressiveness*, as they permit to consider quantitative objectives in a *time frame* context. This addresses a desired wish for many practical applications.

Furthermore, window objectives constitute an attractive alternative in terms of *tractability*. In Chap. 9, we provided algorithms and optimal complexity bounds for one-dimension games. We notably argued that the fixed window variant can be solved in *polynomial time*, which is not known to be the case for the mean-payoff and total-payoff objectives [ZP96, Jur98, GS09, BCD+11].

In Chap. 10, we studied the multi-dimension setting. Fixed window games hold an interesting position. While the associated decision problem is easier to solve than the worst-case mean-payoff threshold problem in one-dimension (P instead of UP ∩ coUP), it becomes comparatively harder in multi-dimension (PSPACE-hard even for polynomial windows instead of coNP). However, it remains EXPTIME-complete for arbitrary windows, in contrast to the total-payoff which becomes undecidable. In terms of complexity, the problem stands in an interesting middle ground between mean-payoff and total-payoff objectives. For the specific case of polynomial windows, there remains a gap between our exponential-time algorithm and the PSPACE lower bound.

Open problem 3. *Can we obtain PSPACE-membership or EXPTIME-hardness for the fixed polynomial window problem in multi-dimension games?*

Filling this small gap would be interesting from a theoretical point of view and witness a fundamental difference (or not) between polynomial and arbitrary windows in the multi-dimension setting.

Still in Chap. 10, we also established a prohibitive lower bound on the complexity of multi-dimension bounded window games: they are at least non-

primitive recursive. While this is clearly bad news on the side of applicability, it would still be of theoretical interest to know if those games are decidable or not.

Open problem 4. *Can we prove (un)decidability of multi-dimension bounded window games?*

Observe that techniques used for the undecidability proof of multi-dimension total-payoff games (Thm. 4.8) cannot be extended easily to the bounded window setting. In particular, our reduction to two-counter machines require to "memorize" sums of weights both negatively and positively. In the window context, such sums can only be memorized negatively (i.e., while windows stay open), as positive windows are closed and forgot immediately (this corresponds to so-called resets in Sect. 10.1). This at the very least indicates that an undecidability result would require a different approach.

Beyond worst-case synthesis. In Chap. 11, we paved the way to a new approach, combining worst-case and expected value requirements in what we named the *beyond worst-case synthesis problem*. We believe this setting is adequate for the synthesis of controllers that must ensure strict guarantees under all circumstances, and prove to be more efficient in reasonable conditions: a problem for which few theoretical frameworks exist. See [MT11, BCFK13] for work with similar philosophy but without worst-case guarantees.

We thoroughly studied the beyond worst-case problem in the context of two well-known quantitative measures: mean-payoff and shortest path.

For the mean-payoff, we proved in Chap. 12 that the problem belongs to $\mathrm{NP} \cap \mathrm{coNP}$, matching the complexity of the worst-case threshold problem, which we encompass. Hence, the beyond worst-case setting provides additional modeling power *at no complexity cost* (in terms of problem solving), a surprisingly positive result. There are still open questions linked to the mean-payoff setting, the first one due to our restriction to finite-memory strategies.

Open problem 5. *What is the exact power of infinite-memory strategies in the beyond worst-case mean-payoff setting? Can we obtain algorithms and complexity bounds for this case?*

This question is not trivial as, to the best of our knowledge, some of the underlying technical results we use do not transfer easily to models with infinite state spaces. The other open problem is also limited to infinite-memory strategies: under finite memory, supremum and infimum variants of the mean-payoff

coincide, which does not need to be the case under infinite memory.

Open problem 6. *Is the infinite-memory setting easier to tackle when considering the supremum mean-payoff variant?*

In Chap. 13, we studied the beyond worst-case problem for the shortest path. We proved it becomes harder than the corresponding worst-case threshold problem, going from polynomial to pseudo-polynomial time, with a complementary result of NP-hardness.

Open problem 7. *Can we prove NP-membership of the beyond worst-case shortest path problem?*

A positive answer seems unlikely as in Sect. 13.4, we prove NP-hardness by reduction from the K^{th} *largest subset problem* which is commonly thought to be outside NP because natural certificates for the problem are larger than polynomial [JK78, GJ79].

Both for the mean-payoff and the shortest path, synthesized strategies may require pseudo-polynomial memory, but accept natural, elegant representations, based on states of the game and simple integer counters.

A limitation of our results for the mean-payoff is that synthesized strategies are parameterized by integer values that must be sufficiently large to ensure good properties. However, we rely on technical results that guarantee the existence of such values without giving an explicit characterization that permit direct computation. In practice, this problem does not really impact our techniques. Indeed, as in the case of the exponential bound for the synthesis algorithm in multi-dimension energy games (Chap. 6), theoretical values may be very large while small values do suffice most of the time. In particular, such values may be too large to be tackled efficiently by synthesis tools: hence we should again follow an *incremental approach* (over such parameter values) to generate winning strategies.

14.2 Future Work

To close this thesis, we discuss several promising research directions. There are of course numerous extensions that would be worth studying. We concentrate our presentation on a few complementary axes, all fitting into the *shift from single-criterion quantitative models toward multi-criteria ones.*

Extensions of window objectives. A first potentially interesting extension, already mentioned in Sect. 8.1.1, is the study of *window objectives in a stochastic context*. As a first step, we could consider Markov decision processes and almost-sure semantics, which would adequately fit the problem. Defining relevant problems based on expected value semantics seems challenging in the context of window objectives. Indeed, there is no global value function to optimize over plays, and considering the expectation over windows may not be the most natural choice. Hence, the first effort should be put in adequate formalization of the interesting questions to solve.

A second question to address is the following. In the multi-dimension setting, our definitions of window objectives (Sect. 8.2) are asynchronous: windows on different dimensions are not required to close simultaneously. *Synchronous variants* may be interesting to study but some useful properties are lost in that setting, such as the inductive property on windows. Hence our techniques cannot be extended straightforwardly.

Lastly, conjunction of window objectives with a parity objective would be interesting to consider. Indeed, a similar notion of time bounds on liveness properties was already studied by Chatterjee et al. through the concept of *finitary* winning [CHH09]. Combining a similar approach with our window objectives seems natural.

Games with mixed objectives. In Part II, we studied games with multi-dimension quantitative objectives, and conjunctions of quantitative objectives with a parity condition. To the extent of our knowledge, all existing results on multi-dimension games are only valid for conjunctions of the same kind of objectives (e.g., several mean-payoff objectives) [CDHR10, VR11, VCD+12, CRR12a, CRR13, CDRR13a]. We would like to investigate how to *mix objectives* of different natures into *unified specifications* and synthesize controllers accordingly. For example, we could study games where the objective is a conjunction of mean-payoff and shortest path. Other similar combinations could be considered.

Extensions of the beyond worst-case framework. We believe that the beyond worst-case framework is a powerful one, well-suited for specifications combining the quest of high expected performance with the need for strong worst-case guarantees. We want to build on the results presented in this thesis and consider several extensions of the initial setting.

The first obvious line of work is applying the same problem to other well-known quantitative measures and to more general classes of games (for example decidable classes of games with imperfect information [DDG+10, HPR13]).

A second interesting question is the extension of our results for mean-payoff and shortest path to multi-dimension games. It is already known that multi-dimension games are more complex than one-dimension ones for the worst-case threshold problem alone [CDHR10, VR11]. Hence, a leap in complexity is also to be expected for the beyond worst-case problem.

Given the relevance of the framework for practical applications, it would certainly be worthwhile to develop tool suites supporting it. We could for example build on symblicit implementations recently developed for monotonic Markov decision processes by Bohy et al. [BBR14].

Other rich behavioral models. Beyond worst-case synthesis provides an appropriate framework for building controllers that are efficient and *strongly risk averse*. Other interesting notions of risk-avoidance have been studied in the literature [FKR95, WL99]. Similarly, combining expected value and variance over the outcomes was considered in [MT11, BCFK13].

We believe that other models may also be of interest. In particular, we study the question of *percentile performances* in uncertain environments, which is unexplored. Given a Markov decision process, we want strategies ensuring specifications such as "at least 90% of the plays achieve a performance level l_1 and 50% achieve level $l_2 > l_1$." Such characterizations would be useful in practice to synthesize controllers with well-understood and reliable performance profiles. This problem yields interesting links with the theory of multi-objective probabilistic systems [KP13]. The model of quantiles for Markov decision processes, as studied by Ummels and Baier [UB13], shares some similarities with our question for the specific case of reachability and accumulated reward.

Links outside computer science are also of interest. Economics is interested in strategies (i.e., investor profiles) that ensure both sufficient risk-avoidance and profitable expected return. Mathematical models powerful enough to tackle the previously discussed problems could be an advantage. A related approach to such questions is the concept of *solvency games* introduced by Berger et al. [BKSV08], and extended by Brázdil et al. [BCF+13]. Solvency games provide a framework for the analysis of risk-averse investors trying to avoid bankruptcy.

Closing words. We conclude with a wise advice from the acclaimed author of *The Art of Computer Programming* [Knu97a, Knu97b, Knu98, Knu11].

> *Beware of bugs in the above code; I have only proved it correct, not tried it.*
>
> Donald E. Knuth, *Notes on the van Emde Boas construction of priority deques: An instructive use of recursion.*

Bibliography

[ACH+10] Parosh Aziz Abdulla, Yu-Fang Chen, Lukás Holík, Richard Mayr, and Tomás Vojnar. When simulation meets antichains. In Esparza and Majumdar [EM10], pages 158–174. *Cited in page 93*

[Ack28] Wilhelm Ackermann. Zum hilbertschen aufbau der reellen zahlen. *Mathematische Annalen*, 99(1):118–133, 1928. *Cited in page 165*

[AD90] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In Paterson [Pat90], pages 322–335. *Cited in page 11*

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994. *Cited in page 11*

[AHK02] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002. *Cited in page 4*

[AMP95] Eugene Asarin, Oded Maler, and Amir Pnueli. Symbolic controller synthesis for discrete and timed systems. In *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 1995. *Cited in page 11*

[AMPS98] Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In *Proceedings of the 5th IFAC Conference on System Structure and Control*, pages 469–474. Elsevier, 1998. *Cited in page 11*

[AMSS13] Parosh Aziz Abdulla, Richard Mayr, Arnaud Sangnier, and Jeremy Sproston. Solving parity games on integer vectors. In Pedro R. D'Argenio and Hernán C. Melgratti, editors, *CONCUR*, volume 8052 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 2013. *2 citations in pages 96 and 269*

[Aum64] Robert J. Aumann. Mixed and behavior strategies in infinite extensive games. In *Advances in Game Theory*, volume 52 of *Annals of Mathematical Studies*, pages 627–650. Princeton University Press, 1964. *Cited in page 21*

[BBC+13] Tomás Brázdil, Václav Brozek, Krishnendu Chatterjee, Vojtech Forejt, and Antonín Kucera. Markov decision processes with multiple long-run average objectives. *Logical Methods in Computer Science*, 2013. *Cited in page 21*

[BBC+11] Tomás Brázdil, Václav Brozek, Krishnendu Chatterjee, Vojtech Forejt, and Antonín Kucera. Two views on multiple mean-payoff objectives in Markov decision processes. In *LICS*, pages 33–42. IEEE Computer Society, 2011. *Cited in page 21*

[BBF+12] Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. Acacia+, a tool for LTL synthesis. In Madhusudan and Seshia [MS12], pages 652–657. *Cited in page 98*

[BBFR13] Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, and Jean-François Raskin. Synthesis from LTL specifications with mean-payoff objectives. In Piterman and Smolka [PS13], pages 169–184. *3 citations in pages 89, 98, and 269*

[BBMU12] Patricia Bouyer, Romain Brenguier, Nicolas Markey, and Michael Ummels. Concurrent games with ordered objectives. In Lars Birkedal, editor, *FoSSaCS*, volume 7213 of *Lecture Notes in Computer Science*, pages 301–315. Springer, 2012. *2 citations in pages 12 and 18*

[BBR14] Aaron Bohy, Véronique Bruyère, and Jean-François Raskin. Symblicit algorithms for mean-payoff and shortest path in mono-

tonic Markov decision processes. *CoRR*, abs/1402.1076, 2014. *Cited in page 274*

[BCD+11] Lubos Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2):97–118, 2011. *4 citations in pages 6, 36, 268, and 270*

[BCF+13] Tomás Brázdil, Taolue Chen, Vojtech Forejt, Petr Novotný, and Aistis Simaitis. Solvency markov decision processes with interest. In Anil Seth and Nisheeth K. Vishnoi, editors, *FSTTCS*, volume 24 of *LIPIcs*, pages 487–499. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013. *Cited in page 274*

[BCFK13] Tomás Brázdil, Krishnendu Chatterjee, Vojtech Forejt, and Antonín Kucera. Trading performance for stability in Markov decision processes. In *LICS*, pages 331–340. IEEE Computer Society, 2013. *4 citations in pages 181, 268, 271, and 274*

[BCHJ09] Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, and Barbara Jobstmann. Better quality in synthesis through quantitative objectives. In Bouajjani and Maler [BM09], pages 140–156. *3 citations in pages 6, 50, and 268*

[BCKN12] Tomás Brázdil, Krishnendu Chatterjee, Antonín Kucera, and Petr Novotný. Efficient controller synthesis for consumption games with multiple resource types. In Madhusudan and Seshia [MS12], pages 23–38. *Cited in page 7*

[BDL+06] Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Håkansson, Paul Pettersson, Wang Yi, and Martijn Hendriks. Uppaal 4.0. In *QEST*, pages 125–126. IEEE Computer Society, 2006. *Cited in page 11*

[Bee80] Catriel Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Trans. Database Syst.*, 5(3):241–259, 1980. *3 citations in pages 5, 31, and 138*

[Ber95] Dimitri P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995. *Cited in page 7*

[BFL+08] Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, and Jirí Srba. Infinite runs in weighted timed automata with energy constraints. In Franck Cassez and Claude Jard, editors, *FORMATS*, volume 5215 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2008. *3 citations in pages 6, 37, and 268*

[BFRR13] Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *CoRR*, abs/1309.5439, 2013. 45 pages. *6 citations in pages 9, 24, 44, 177, 185, and 247*

[BFRR14a] Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin. Expectations or guarantees? I want it all! A crossroad between games and MDPs. In *Proc. of SR*, EPTCS, 2014. 7 pages. *5 citations in pages 9, 44, 177, 185, and 247*

[BFRR14b] Véronique Bruyère, Emmanuel Filiot, Mickael Randour, and Jean-François Raskin. Meet Your Expectations With Guarantees: Beyond Worst-Case Synthesis in Quantitative Games. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, volume 25 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 199–213, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. *5 citations in pages 9, 44, 177, 185, and 247*

[BGHJ09] Roderick Bloem, Karin Greimel, Thomas A. Henzinger, and Barbara Jobstmann. Synthesizing robust systems. In *FMCAD*, pages 85–92. IEEE, 2009. *Cited in page 50*

[BHR14] Udi Boker, Thomas A. Henzinger, and Arjun Radhakrishna. Battery transition systems. In Suresh Jagannathan and Peter Sewell, editors, *POPL*, pages 595–606. ACM, 2014. *Cited in page 7*

[BJK10] Tomás Brázdil, Petr Jancar, and Antonín Kucera. Reachability games on extended vector addition systems with states. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP (2)*, volume 6199 of *Lecture Notes in Computer Science*, pages 478–489. Springer, 2010.
12 citations in pages 41, 50, 54, 69, 72, 73, 74, 75, 76, 77, 95, and 269

[BJW02] Julien Bernet, David Janin, and Igor Walukiewicz. Permissive strategies: from parity games to safety games. *ITA*, 36(3):261–275, 2002. *Cited in page 78*

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.
7 citations in pages 2, 7, 22, 31, 105, 110, and 182

[BKSV08] Noam Berger, Nevin Kapur, Leonard J. Schulman, and Vijay V. Vazirani. Solvency games. In Ramesh Hariharan, Madhavan Mukund, and V. Vinay, editors, *FSTTCS*, volume 2 of *LIPIcs*, pages 61–72. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2008.
Cited in page 274

[BM09] Ahmed Bouajjani and Oded Maler, editors. *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, volume 5643 of *Lecture Notes in Computer Science*. Springer, 2009. *2 citations in pages 279 and 291*

[BMOU11] Patricia Bouyer, Nicolas Markey, Jörg Olschewski, and Michael Ummels. Measuring permissiveness in parity games: Mean-payoff parity games revisited. In Tevfik Bultan and Pao-Ann Hsiung, editors, *ATVA*, volume 6996 of *Lecture Notes in Computer Science*, pages 135–149. Springer, 2011.
7 citations in pages 7, 30, 49, 50, 66, 110, and 268

[BMR14] Véronique Bruyère, Noémie Meunier, and Jean-François Raskin. Secure equilibria in weighted games. *CoRR*, abs/1402.3962, 2014.
Cited in page 18

[Bre13] Romain Brenguier. *Nash Equilibria in Concurrent Games – Application to Timed Games*. PhD thesis, Laboratoire Spécification et Vérification, ENS Cachan, France, 2013. *Cited in page 11*

[Bri06] Thomas Brihaye. *Verification and Control of O-Minimal Hybrid Systems and Weighted Timed Automata*. PhD thesis, Université de Mons-Hainaut, Belgium, 2006. *Cited in page 11*

[BT76] Itshak Borosh and Leon B. Treybig. Bounds on positive integral solutions of linear diophantine equations. *Proc. of the American Mathematical Society*, 55(2):299–304, 1976. *Cited in page 73*

[BT91] Dimitri P. Bertsekas and John N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16:580–595, 1991. *2 citations in pages 8 and 36*

[Büc60] Julius Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960. *Cited in page 3*

[BV38] Emile Borel and Jean Ville. *Applications aux jeux de hasard*. Gauthier-Vilars, 1938. *Cited in page 3*

[BV07] Henrik Björklund and Sergei G. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155(2):210–229, 2007. *2 citations in pages 6 and 36*

[CCH+11] Pavol Cerný, Krishnendu Chatterjee, Thomas A. Henzinger, Arjun Radhakrishna, and Rohit Singh. Quantitative synthesis for concurrent programs. In Gopalakrishnan and Qadeer [GQ11], pages 243–259. *Cited in page 50*

[CD11] Krishnendu Chatterjee and Laurent Doyen. Games and Markov decision processes with mean-payoff parity and energy parity objectives. In Zdenek Kotásek, Jan Bouda, Ivana Cerná, Lukás Sekanina, Tomás Vojnar, and David Antos, editors, *MEMICS*, volume 7119 of *Lecture Notes in Computer Science*, pages 37–46. Springer, 2011. *Cited in page 30*

[CD12] Krishnendu Chatterjee and Laurent Doyen. Energy parity games. *Theor. Comput. Sci.*, 458:49–60, 2012.
9 citations in pages 7, 49, 50, 66, 70, 85, 107, 108, and 268

[CdAHS03] Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Mariëlle Stoelinga. Resource interfaces. In Rajeev Alur and Insup Lee, editors, *EMSOFT*, volume 2855 of *Lecture Notes in Computer Science*, pages 117–133. Springer, 2003.
3 citations in pages 6, 37, and 268

[CDGO14] Krishnendu Chatterjee, Laurent Doyen, Hugo Gimbert, and Youssouf Oualhadj. Perfect-information stochastic mean-payoff parity games. In *FOSSACS*, Lecture Notes in Computer Science. Springer, 2014. *Cited in page 12*

[CDH09] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. A survey of stochastic games with limsup and liminf objectives. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *ICALP (2)*, volume 5556 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2009. *Cited in page 28*

[CDH10a] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log.*, 11(4), 2010. *Cited in page 50*

[CDH10b] Julien Cristau, Claire David, and Florian Horn. How do we remember the past in randomised strategies? In Angelo Montanari, Margherita Napoli, and Mimmo Parente, editors, *GANDALF*, volume 25 of *EPTCS*, pages 30–39, 2010. *Cited in page 21*

[CDHR06] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Algorithms for omega-regular games with imperfect information. In Zoltán Ésik, editor, *CSL*, volume 4207 of *Lecture Notes in Computer Science*, pages 287–302. Springer, 2006. *Cited in page 12*

[CDHR10] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Generalized mean-payoff and energy games. In Kamal Lodaya and Meena Mahajan, editors, *FSTTCS*, volume 8 of *LIPIcs*, pages 505–516. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.
8 citations in pages 7, 40, 50, 53, 70, 268, 273, and 274

[CDRR13a] Krishnendu Chatterjee, Laurent Doyen, Mickael Randour, and Jean-François Raskin. Looking at mean-payoff and total-payoff through windows. In Hung and Ogawa [HO13], pages 118–132.
11 citations in pages 9, 40, 42, 49, 51, 58, 60, 119, 131, 151, and 273

[CDRR13b] Krishnendu Chatterjee, Laurent Doyen, Mickael Randour, and Jean-François Raskin. Looking at mean-payoff and total-payoff through windows. *CoRR*, abs/1302.4248, 2013. 30 pages.
10 citations in pages 9, 40, 42, 49, 51, 58, 60, 119, 131, and 151

[CE81] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In Dexter Kozen, editor, *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.
2 citations in pages 2 and 80

[CFK+12] Taolue Chen, Vojtech Forejt, Marta Z. Kwiatkowska, Aistis Simaitis, Ashutosh Trivedi, and Michael Ummels. Playing stochastic games precisely. In Koutny and Ulidowski [KU12], pages 348–363.
Cited in page 21

[CFK+13a] Taolue Chen, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, and Aistis Simaitis. PRISM-games: A model checker for stochastic multi-player games. In Piterman and Smolka [PS13], pages 185–191.
2 citations in pages 12 and 21

[CFK+13b] Taolue Chen, Vojtech Forejt, Marta Z. Kwiatkowska, Aistis Simaitis, and Clemens Wiltsche. On stochastic games with multiple objectives. In Krishnendu Chatterjee and Jiri Sgall, editors, *MFCS*, volume 8087 of *Lecture Notes in Computer Science*, pages 266–277. Springer, 2013. *2 citations in pages 12 and 28*

[CGP00] Edmund M. Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT Press, 2000. *Cited in page 2*

[CGR96] Boris V. Cherkassky, Andrew V. Goldberg, and Tomasz Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical programming*, 73(2):129–174, 1996. *Cited in page 36*

[CH11] Krishnendu Chatterjee and Monika Henzinger. Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification. In Dana Randall, editor, *SODA*, pages 1318–1336. SIAM, 2011. *Cited in page 31*

[CH12] Krishnendu Chatterjee and Monika Henzinger. An $O(n^2)$ time algorithm for alternating Büchi games. In Yuval Rabani, editor, *SODA*, pages 1386–1399. SIAM, 2012.
8 citations in pages 6, 7, 32, 154, 155, 190, 205, and 208

[Cha07a] Krishnendu Chatterjee. Concurrent games with tail objectives. *Theor. Comput. Sci.*, 388(1-3):181–198, 2007.
2 citations in pages 12 and 18

[Cha07b] Krishnendu Chatterjee. *Stochastic ω-Regular Games*. PhD thesis, University of California at Berkeley, 2007. *Cited in page 12*

[CHH09] Krishnendu Chatterjee, Thomas A. Henzinger, and Florian Horn. Finitary winning in omega-regular games. *ACM Trans. Comput. Log.*, 11(1), 2009. *Cited in page 273*

[CHJ04] Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdziński. Games with secure equilibria. In *LICS*, pages 160–169. IEEE Computer Society, 2004. *Cited in page 18*

[CHJ05] Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdziński. Mean-payoff parity games. In *LICS*, pages 178–187. IEEE Computer Society, 2005.
8 citations in pages 7, 30, 49, 50, 66, 107, 112, and 268

[CHKN12] Krishnendu Chatterjee, Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Polynomial-time algorithms for energy

games with special weight structures. In Leah Epstein and Paolo Ferragina, editors, *ESA*, volume 7501 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2012. *Cited in page 6*

[CHR12] Pavol Cerný, Thomas A. Henzinger, and Arjun Radhakrishna. Simulation distances. *Theor. Comput. Sci.*, 413(1):21–35, 2012. *Cited in page 50*

[Chu57] Alonzo Church. Applications of recursive arithmetic to the problem of circuit synthesis. *Summaries of the Summer Institute of Symbolic Logic*, 1:3–50, 1957. *Cited in page 3*

[Chu62] Alonzo Church. Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35. Institut Mittag-Leffler, 1962. *Cited in page 4*

[CJH03] Krishnendu Chatterjee, Marcin Jurdziński, and Thomas A. Henzinger. Simple stochastic parity games. In Matthias Baaz and Johann A. Makowsky, editors, *CSL*, volume 2803 of *Lecture Notes in Computer Science*, pages 100–113. Springer, 2003. *2 citations in pages 7 and 31*

[CJH04] Krishnendu Chatterjee, Marcin Jurdziński, and Thomas A. Henzinger. Quantitative stochastic parity games. In J. Ian Munro, editor, *SODA*, pages 121–130. SIAM, 2004. *2 citations in pages 7 and 33*

[CKS81] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981. *4 citations in pages 44, 155, 156, and 157*

[CMH08] Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. Stochastic limit-average games are in EXPTIME. *Int. J. Game Theory*, 37(2):219–234, 2008. *Cited in page 28*

[Con93] Anne Condon. On algorithms for simple stochastic games. *Advances in computational complexity theory*, 13:51–73, 1993. *Cited in page 36*

[Cou38] Antoine-Augustin Cournot. *Recherches sur les principes mathématiques de la théorie des richesses*. L. Hachette, 1838.
Cited in page 3

[CRR12a] Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. In Koutny and Ulidowski [KU12], pages 115–131.
9 citations in pages 8, 40, 51, 66, 69, 89, 98, 99, and 273

[CRR12b] Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *CoRR*, abs/1201.5073, 2012. 29 pages.
8 citations in pages 8, 40, 51, 66, 69, 89, 98, and 99

[CRR13] Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 2013. 35 pages.
9 citations in pages 8, 40, 51, 66, 69, 89, 98, 99, and 273

[CY90] Costas Courcoubetis and Mihalis Yannakakis. Markov decision processes and regular events. In Paterson [Pat90], pages 336–349.
2 citations in pages 7 and 33

[CY95] Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.
2 citations in pages 188 and 201

[dA97] Luca de Alfaro. *Formal verification of probabilistic systems*. PhD thesis, Stanford University, 1997.
5 citations in pages 7, 33, 182, 188, and 201

[dA99] Luca de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In Jos C. M. Baeten and Sjouke Mauw, editors, *CONCUR*, volume 1664 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 1999.
2 citations in pages 8 and 36

[dAH01] Luca de Alfaro and Thomas A. Henzinger. Interface theories for component-based design. In Thomas A. Henzinger and Christoph M. Kirsch, editors, *EMSOFT*, volume 2211 of *Lecture Notes in Computer Science*, pages 148–165. Springer, 2001.
Cited in page 4

[dAHK98] Luca de Alfaro, Thomas A. Henzinger, and Orna Kupferman. Concurrent reachability games. In *FOCS*, pages 564–575. IEEE Computer Society, 1998. *2 citations in pages 12 and 18*

[DDG+10] Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Torunczyk. Energy and mean-payoff games with imperfect information. In Anuj Dawar and Helmut Veith, editors, *CSL*, volume 6247 of *Lecture Notes in Computer Science*, pages 260–274. Springer, 2010. *2 citations in pages 12 and 274*

[DDHR06] Martin De Wulf, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Antichains: A new algorithm for checking universality of finite automata. In Thomas Ball and Robert B. Jones, editors, *CAV*, volume 4144 of *Lecture Notes in Computer Science*, pages 17–30. Springer, 2006. *3 citations in pages 9, 41, and 93*

[De 13] Julie De Pril. *Equilibria in Multiplayer Cost Game*. PhD thesis, University of Mons, Belgium, 2013. *2 citations in pages 11 and 18*

[DFS98] Catherine Dufourd, Alain Finkel, and Philippe Schnoebelen. Reset nets between decidability and undecidability. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115. Springer, 1998. *Cited in page 165*

[Dic13] Leonard E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *American Journal of Mathematics*, 35(4):413–422, 1913. *Cited in page 54*

[DR10] Laurent Doyen and Jean-François Raskin. Antichain algorithms for finite automata. In Esparza and Majumdar [EM10], pages 2–22.
2 citations in pages 89 and 93

[DR11] Laurent Doyen and Jean-François Raskin. Games with imperfect information: Theory and algorithms. In *Lectures in Game Theory for Computer Scientists*, pages 185–212. Cambridge University Press, 2011. *Cited in page 78*

[Duc13] Marc Ducobu. *Antichains for QBF evaluation and VPA decision problems.* PhD thesis, University of Mons, Belgium, 2013. *Cited in page 93*

[EJ88] E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs. In *FOCS*, pages 328–337. IEEE Computer Society, 1988. *2 citations in pages 6 and 33*

[EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS*, pages 368–377. IEEE Computer Society, 1991. *3 citations in pages 6, 32, and 164*

[EJS93] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model-checking for fragments of μ-calculus. In Costas Courcoubetis, editor, *CAV*, volume 697 of *Lecture Notes in Computer Science*, pages 385–396. Springer, 1993. *2 citations in pages 6 and 33*

[EKVY08] Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-objective model checking of Markov decision processes. *Logical Methods in Computer Science*, 4(4), 2008. *Cited in page 12*

[EM79] Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *Int. Journal of Game Theory*, 8(2):109–113, 1979. *5 citations in pages 6, 35, 58, 198, and 268*

[EM10] Javier Esparza and Rupak Majumdar, editors. *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science*. Springer, 2010. *2 citations in pages 277 and 288*

[Esp96] Javier Esparza. Decidability and complexity of petri net problems - an introduction. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*, pages 374–428. Springer, 1996. *Cited in page 165*

[Eve57] Hugh Everett. Recursive games. *Contributions to the Theory of Games*, 3(39):47–78, 1957. *Cited in page 12*

[FH10] Nathanaël Fijalkow and Florian Horn. The surprizing complexity of generalized reachability games. *CoRR*, abs/1010.2420, 2010. *4 citations in pages 44, 162, 163, and 164*

[FJLS11] Uli Fahrenberg, Line Juhl, Kim G. Larsen, and Jirí Srba. Energy games in multiweighted automata. In Antonio Cerone and Pekka Pihlajasaari, editors, *ICTAC*, volume 6916 of *Lecture Notes in Computer Science*, pages 95–115. Springer, 2011. *3 citations in pages 50, 95, and 269*

[FKR95] Jerzy A. Filar, Dmitry Krass, and Kirsten W. Ross. Percentile performance criteria for limiting average Markov decision processes. *Transactions on Automatic Control*, 40(1):2–10, 1995. *2 citations in pages 180 and 274*

[FV97] Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer, 1997. *5 citations in pages 8, 34, 36, 58, and 188*

[Gil57] Dean Gillette. Stochastic games with zero stop probabilities. *Contributions to the Theory of Games*, 3:179–187, 1957. *Cited in page 6*

[Gim07] Hugo Gimbert. Pure stationary optimal strategies in Markov decision processes. In Wolfgang Thomas and Pascal Weil, editors, *STACS*, volume 4393 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 2007. *2 citations in pages 8 and 36*

[GJ79] Michael R. Garey and David S. Johnson. *Computers and intractability: a guide to the Theory of NP-Completeness*. Freeman New York, 1979. *4 citations in pages 45, 256, 257, and 272*

[GKK88] Vladimir A. Gurvich, Alexander V. Karzanov, and L.G. Khachivan. Cyclic games and an algorithm to find minimax cycle means in directed graphs. *USSR Computational Mathematics and Mathematical Physics*, 28(5):85–91, 1988. *2 citations in pages 6 and 36*

[GO02] Peter W. Glynn and Dirk Ormoneit. Hoeffding's inequality for uniformly ergodic Markov chains. *Statistics & Probability Letters*, 56(2):143–146, 2002. *3 citations in pages 193, 216, and 217*

[GQ11] Ganesh Gopalakrishnan and Shaz Qadeer, editors. *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*. Springer, 2011. *2 citations in pages 282 and 293*

[GS97] Charles M. Grinstead and J. Laurie Snell. *Introduction to probability*. American Mathematical Society, 1997. *4 citations in pages 105, 193, 239, and 243*

[GS09] Thomas Gawlitza and Helmut Seidl. Games through nested fixpoints. In Bouajjani and Maler [BM09], pages 291–305. *8 citations in pages 6, 34, 35, 58, 130, 142, 268, and 270*

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. *2 citations in pages 3 and 5*

[GU08] Erich Grädel and Michael Ummels. Solution Concepts and Algorithms for Infinite Multiplayer Games. In Krzysztof Apt and Robert van Rooij, editors, *New Perspectives on Games and Interaction*, volume 4 of *Texts in Logic and Games*, pages 151–178. Amsterdam University Press, 2008. *2 citations in pages 11 and 18*

[GZ04] Hugo Gimbert and Wieslaw Zielonka. When can you play positionally? In Jirí Fiala, Václav Koubek, and Jan Kratochvíl, editors, *MFCS*, volume 3153 of *Lecture Notes in Computer Science*, pages 686–697. Springer, 2004. *4 citations in pages 6, 34, 58, and 143*

[Hen96] Thomas A. Henzinger. The theory of hybrid automata. In *LICS*, pages 278–292. IEEE Computer Society, 1996. *Cited in page 11*

[HHWT97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. In Orna Grumberg, editor, *CAV*, volume 1254 of *Lecture Notes in Computer Science*, pages 460–463. Springer, 1997. *Cited in page 11*

[HKR02] Thomas A. Henzinger, Orna Kupferman, and Sriram K. Rajamani. Fair simulation. *Inf. Comput.*, 173(1):64–81, 2002. *Cited in page 4*

[HO13] Dang Van Hung and Mizuhito Ogawa, editors. *Automated Technology for Verification and Analysis - 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15-18, 2013. Proceedings*, volume 8172 of *Lecture Notes in Computer Science*. Springer, 2013. *2 citations in pages 284 and 294*

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. *Cited in page 216*

[HP85] David Harel and Amir Pnueli. On the development of reactive systems. In Krzysztof R. Apt, editor, *Logics and Models of Concurrent Systems*, volume 13 of *NATO ASI Series*, pages 477–498. Springer Berlin Heidelberg, 1985. *Cited in page 2*

[HPR13] Paul Hunter, Guillermo A. Pérez, and Jean-François Raskin. Mean-payoff games with incomplete information. *CoRR*, abs/1309.5462, 2013. *2 citations in pages 12 and 274*

[Imm81] Neil Immerman. Number of quantifiers is better than number of tape cells. *J. Comput. Syst. Sci.*, 22(3):384–406, 1981. *3 citations in pages 5, 31, and 138*

[JK78] Donald B. Johnson and Samuel D. Kashdan. Lower bounds for selection in X + Y and other multisets. *Journal of the ACM*, 25(4):556–570, 1978. *4 citations in pages 45, 256, 257, and 272*

[JSL08] Marcin Jurdziński, Jeremy Sproston, and François Laroussinie. Model checking probabilistic timed automata with one or two clocks. *Logical Methods in Computer Science*, 4(3), 2008.
4 citations in pages 44, 155, 160, and 161

[Jur98] Marcin Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998.
6 citations in pages 6, 33, 35, 58, 268, and 270

[KKV01] Valerie King, Orna Kupferman, and Moshe Y. Vardi. On the complexity of parity word automata. In Furio Honsell and Marino Miculan, editors, *FoSSaCS*, volume 2030 of *Lecture Notes in Computer Science*, pages 276–286. Springer, 2001.
2 citations in pages 6 and 33

[KL93] Alexander V. Karzanov and Vasilij N. Lebedev. Cyclical games with prohibitions. *Math. Program.*, 60:277–293, 1993. *Cited in page 35*

[KLST12] Miroslav Klimos, Kim G. Larsen, Filip Stefanak, and Jeppe Thaarup. Nash equilibria in concurrent priced games. In Adrian Horia Dediu and Carlos Martín-Vide, editors, *LATA*, volume 7183 of *Lecture Notes in Computer Science*, pages 363–376. Springer, 2012.
2 citations in pages 12 and 18

[KNP11] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In Gopalakrishnan and Qadeer [GQ11], pages 585–591. *Cited in page 12*

[Knu97a] Donald E. Knuth. *The Art of Computer Programming, Volume 1: Fundamental Algorithms, 3rd Edition.* Addison-Wesley, 1997.
Cited in page 275

[Knu97b] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms, 3rd Edition.* Addison-Wesley, 1997.
Cited in page 275

[Knu98] Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching, 2nd Edition.* Addison-Wesley, 1998.
Cited in page 275

[Knu11] Donald E. Knuth. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1.* Addison-Wesley, 2011. *Cited in page 275*

[Kop06] Eryk Kopczynski. Half-positional determinacy of infinite games. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 336–347. Springer, 2006. *Cited in page 55*

[KP13] Marta Z. Kwiatkowska and David Parker. Automated verification and strategy synthesis for probabilistic systems. In Hung and Ogawa [HO13], pages 5–22. *2 citations in pages 12 and 274*

[KS88] S. Rao Kosaraju and Gregory F. Sullivan. Detecting cycles in dynamic graphs in polynomial time. In Janos Simon, editor, *STOC*, pages 398–406. ACM, 1988. *Cited in page 55*

[KU12] Maciej Koutny and Irek Ulidowski, editors. *CONCUR 2012 - Concurrency Theory - 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings*, volume 7454 of *Lecture Notes in Computer Science*. Springer, 2012. *2 citations in pages 284 and 287*

[Kő36] Dénes Kőnig. *Theorie der endlichen und unendlichen Graphen.* Akademische Verlagsgesellschaft, Leipzig, 1936. *Cited in page 54*

[LL69] Thomas M. Liggett. and Steven A. Lippman. Stochastic games with perfect information and time average payoff. *Siam Review*, 11(4):604–607, 1969. *4 citations in pages 6, 12, 35, and 58*

[LNO+08] Ranko Lazic, Tom Newcomb, Joël Ouaknine, A. W. Roscoe, and James Worrell. Nets with tokens which carry data. *Fundam. Inform.*, 88(3):251–274, 2008. *3 citations in pages 44, 165, and 167*

[LP07] Yuri M. Lifshits and Dmitri S. Pavlov. Potential theory for mean payoff games. *Journal of Mathematical Sciences*, 145(3):4967–4974, 2007. *2 citations in pages 6 and 36*

[Maq11] Nicolas Maquet. *New Algorithms and Data Structures for the Emptiness Problem of Alternating Automata.* PhD thesis, Université Libre de Bruxelles, Belgium, 2011. *Cited in page 93*

[Mar75] Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975. *4 citations in pages 5, 29, 125, and 198*

[Mar98] Donald A. Martin. The determinacy of Blackwell games. *J. Symb. Log.*, 63(4):1565–1581, 1998. *3 citations in pages 5, 29, and 125*

[McN93] Robert McNaughton. Infinite games played on finite graphs. *Ann. Pure Appl. Logic*, 65(2):149–184, 1993. *2 citations in pages 6 and 32*

[Min61] Marvin L. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *The Annals of Mathematics*, 74(3):437–455, 1961. *2 citations in pages 40 and 62*

[MN81] Jean-François Mertens and Abraham Neyman. Stochastic games. *International Journal of Game Theory*, 10(2):53–66, 1981. *Cited in page 12*

[MS12] P. Madhusudan and Sanjit A. Seshia, editors. *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*, volume 7358 of *Lecture Notes in Computer Science*. Springer, 2012. *2 citations in pages 278 and 279*

[MT11] Shie Mannor and John N. Tsitsiklis. Mean-variance optimization in Markov decision processes. In Lise Getoor and Tobias Scheffer, editors, *ICML*, pages 177–184. Omnipress, 2011. *4 citations in pages 181, 268, 271, and 274*

[Nas50] John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49, 1950. *Cited in page 18*

[OR94] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory.* MIT Press, 1994. *2 citations in pages 3 and 17*

[Pap94] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994. *4 citations in pages 5, 6, 33, and 55*

[Pat90] Mike Paterson, editor. *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, July 16-20, 1990, Proceedings*, volume 443 of *Lecture Notes in Computer Science*. Springer, 1990. *2 citations in pages 277 and 287*

[Pis99] Nicolai N. Pisaruk. Mean cost cyclical games. *Mathematics of Operations Research*, 24(4):817–828, 1999. *2 citations in pages 6 and 36*

[Pnu77] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE Computer Society, 1977. *2 citations in pages 31 and 98*

[PR89] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190. ACM Press, 1989. *2 citations in pages 3 and 4*

[PS13] Nir Piterman and Scott A. Smolka, editors. *Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, volume 7795 of *Lecture Notes in Computer Science*. Springer, 2013. *2 citations in pages 278 and 284*

[Put94] Martin L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994. *2 citations in pages 8 and 36*

[Rab69] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969. *Cited in page 3*

[Rac78] Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theor. Comput. Sci.*, 6:223–231, 1978. *Cited in page 71*

[Ran13] Mickael Randour. Automated synthesis of reliable and efficient systems through game theory: A case study. In Thomas Gilbert, Markus Kirkilionis, and Gregoire Nicolis, editors, *Proceedings of the European Conference on Complex Systems 2012*, Springer Proceedings in Complexity, pages 731–738. Springer, 2013. *Cited in page 1*

[Rei84] John H. Reif. The complexity of two-player games of incomplete information. *Journal of computer and system sciences*, 29(2):274–301, 1984. *Cited in page 12*

[RW87] Peter J. Ramadge and W. Murray Wonham. Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization*, 25(1):206–230, 1987. *2 citations in pages 3 and 4*

[RY86] Louis E. Rosier and Hsu-Chun Yen. A multiparameter analysis of the boundedness problem for vector addition systems. *J. Comput. Syst. Sci.*, 32(1):105–135, 1986. *Cited in page 71*

[San13] Ocan Sankur. *Robustness in Timed Automata: Analysis, Synthesis, Implementation*. PhD thesis, Laboratoire Spécification et Vérification, ENS Cachan, France, 2013. *Cited in page 11*

[Sch02] Philippe Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Inf. Process. Lett.*, 83(5):251–261, 2002. *3 citations in pages 44, 165, and 167*

[Sha53] Lloyd S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 39(10):1095, 1953. *Cited in page 12*

[Sip97] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997. *Cited in page 5*

[Sol03] Eilon Solan. Continuity of the value of competitive Markov decision processes. *Journal of Theoretical Probability*, 16(4):831–845, 2003. *Cited in page 111*

[Tho95] Wolfgang Thomas. On the synthesis of strategies in infinite games. In *STACS*, pages 1–13, 1995. *Cited in page 3*

[Tho97] Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, volume 3, Beyond Words, chapter 7, pages 389–455. Springer, 1997. *3 citations in pages 4, 6, and 32*

[Tra09] Mathieu Tracol. Fast convergence to state-action frequency polytopes for MDPs. *Oper. Res. Lett.*, 37(2):123–126, 2009. *3 citations in pages 193, 216, and 217*

[TV87] Frank Thuijsman and Okko Jan Vrieze. The bad match; a total reward stochastic game. *Operations-Research-Spektrum*, 9(2):93–99, 1987. *Cited in page 12*

[UB13] Michael Ummels and Christel Baier. Computing quantiles in markov reward models. In Frank Pfenning, editor, *FoSSaCS*, volume 7794 of *Lecture Notes in Computer Science*, pages 353–368. Springer, 2013. *Cited in page 274*

[Umm10] Michael Ummels. *Stochastic Multiplayer Games: Theory and Algorithms*. PhD thesis, RWTH Aachen University, 2010. *Cited in page 11*

[Var85] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS*, pages 327–338. IEEE Computer Society, 1985. *Cited in page 22*

[VCD+12] Yaron Velner, Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, Alexander Rabinovich, and Jean-François Raskin. The complexity of multi-mean-payoff and multi-energy games. *CoRR*, abs/1209.3234, 2012. *12 citations in pages 7, 40, 49, 50, 53, 54, 55, 56, 57, 60, 65, and 273*

[vN28] John von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928. *Cited in page 17*

[vNM44] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944. *Cited in page 3*

[VR11] Yaron Velner and Alexander Rabinovich. Church synthesis problem for noisy input. In Martin Hofmann, editor, *FOSSACS*, volume 6604 of *Lecture Notes in Computer Science*, pages 275–289. Springer, 2011. *7 citations in pages 7, 40, 50, 53, 268, 273, and 274*

[VW86] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *LICS*, pages 332–344. IEEE Computer Society, 1986. *Cited in page 2*

[WL99] Congbin Wu and Yuanlie Lin. Minimizing risk models in Markov decision processes with policies depending on target values. *Journal of Mathematical Analysis and Applications*, 231(1):47–67, 1999. *2 citations in pages 180 and 274*

[Zer13] Ernst Zermelo. Über eine anwendung der mengenlehre auf die theorie des schachspiels. In *Proceedings of the fifth international congress of mathematicians*, volume 2, pages 501–504. II, Cambridge UP, Cambridge, 1913. *Cited in page 3*

[Zie98] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998. *3 citations in pages 6, 32, and 33*

[ZP96] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158(1&2):343–359, 1996. *6 citations in pages 6, 35, 36, 58, 268, and 270*

Index

Glossary of Notations

Graphs

$\mathcal{G} = (S, E, w)$	(weighted) directed graph
S	set of states
$s \in S$	state
$s_{\mathsf{init}} \in S$	initial state
$\mathsf{Succ}(s) \subseteq S$	successors of state s
$E \subseteq S \times S$	set of edges
$e \in E$	edge
$w \colon E \to \mathbb{Z}$	integer weight function
k	dimension of weight vectors
$w \colon E \to \mathbb{Z}^k$	integer vector weight function
W	largest absolute weight on any edge
$V = \lceil \log_2 W \rceil$	length of the binary encoding of W
$\mathcal{G} \restriction A$	subgraph induced by $A \subseteq S$

Plays

$\pi = s_0 s_1 s_2 \ldots \in S^\omega$	play
$\pi(n) = s_0 s_1 \ldots s_n$	prefix of π up to the n-th state
ρ	prefix
$\pi(n, \infty)$	infinite suffix starting in s_n
$\mathsf{First}(\pi(n)) = s_0$	first state of $\pi(n)$
$\mathsf{First}(\pi) = s_0$	first state of π
$\mathsf{Last}(\pi(n)) = s_n$	last state of $\pi(n)$

$\mathsf{Plays}(\mathcal{G})$	set of plays of graph $\mathcal{G}$
$\mathsf{Prefs}(\mathcal{G})$	set of prefixes of graph $\mathcal{G}$
$\mathsf{Inf}(\pi) \subseteq S$	set of states visited infinitely often along play π
$p\colon S \to \mathbb{N}$	priority function
$\mathsf{Par}(\pi)$	parity of play π

Games

$\mathcal{P}_1$	first player (i.e., the controller)
$\mathcal{P}_2$	second player (i.e., the environment)
$G = (\mathcal{G}, S_1, S_2)$	game
$G = (S_1, S_2, E, w)$	game (w/o graph reference)
$\mathsf{Prefs}_i(G)$	prefixes belonging to $\mathcal{P}_i$
$\lvert G \rvert$	size of the game
$G \restriction A$	subgame induced by $A \subseteq S$
$G = (S_1, S_2, E, k, w)$	multi-dimension game
$G_p = (S_1, S_2, E, k, w, p)$	multi-dimension game with parity objective
G_Δ	games reduced to edges in E_Δ

Probabilities

$d\colon A \to [0,1] \cap \mathbb{Q}$	probability distribution over A
$\mathcal{D}(A)$	set of all probability distributions over A
$\mathsf{Supp}(d)$	support of probability distribution d

Strategies

λ_i	strategy (of $\mathcal{P}_i$)
$\mathcal{M}(\lambda_i^f)$	SOMM encoding finite-memory strategy λ_i^f
Λ_i	set of general strategies (for $\mathcal{P}_i$)
Λ_i^F	set of finite-memory strategies
Λ_i^{PF}	set of pure finite-memory strategies
Λ_i^M	set of memoryless strategies
Λ_i^{PM}	set of pure memoryless strategies
Λ_i^{RM}	set of randomized memoryless strategies

Strategies for BWC synthesis

$\lambda_2^{\mathsf{stoch}}$	stochastic model of the environment
λ_1^{wc}	worst-case winning strategy
λ_1^{e}	optimal expectation strategy
λ_1^{cmb}	combined strategy (for WECs with $E_\Delta = E$)
λ_1^{wns}	witness-and-secure strategy (for all WECs)
λ_1^{glb}	global strategy
L	steps for worst-case strategy in combined one
K	steps for expectation strategy in combined one
N	steps for expectation strategy in global one

Markov Decision Processes

$P = (\mathcal{G}, S_1, S_\Delta, \Delta)$	Markov decision process
$P = (S_1, S_\Delta, E, \Delta, w)$	Markov decision process (w/o graph reference)
S_Δ	set of stochastic states
$\Delta \colon S_\Delta \to \mathcal{D}(S)$	stochastic transition function
$G[\lambda_i]$	MDP resulting from fixing λ_i in game G
$P \restriction A$	sub-MDP induced by $A \subseteq S$
$U \subseteq S$	end-component
$\mathcal{E} \subseteq 2^S$	set of all end-components
$E_\Delta \subseteq E$	edges with non-zero probability
P_Δ	MDP reduced to edges in E_Δ
$\mathcal{W} \subseteq \mathcal{E}$	set of winning end-components
$\mathcal{U}_{\mathrm{W}} \subseteq \mathcal{E}$	set of maximal winning end-components
$\mathcal{L} \subseteq \mathcal{E}$	set of losing end-components

Markov Chains

$M = (\mathcal{G}, \delta)$	Markov chain
$M = (S, E, \delta, w)$	Markov chain (w/o graph reference)
$\delta \colon S \to \mathcal{D}(S)$	stochastic transition function
$\mathcal{A} \subseteq \mathsf{Plays}(\mathcal{G})$	event (measurable set of plays)
$\mathbb{P}^M_{s_{\mathsf{init}}}(\mathcal{A})$	probability measure of event $\mathcal{A}$
$\mathbb{E}^M_{s_{\mathsf{init}}}(f)$	expected value of f

$G[\lambda_1, \lambda_2]$	MC resulting from fixing λ_1 and λ_2 in game G
$P[\lambda_1]$	MC resulting from fixing λ_1 in MDP P

Outcomes

$\mathsf{Outs}_M(s_{\mathsf{init}})$	outcomes of MC M starting in s_{init}
$\mathsf{Outs}_P(s_{\mathsf{init}}, \lambda_1)$	outcomes of MDP P under strategy λ_1
$\mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_1, \lambda_2)$	outcomes of game G under strategies λ_1 and λ_2
$\mathsf{Outs}_G(s_{\mathsf{init}}, \lambda_i)$	outcomes of game G consistent with λ_i

Value Functions and Classical Objectives

f	value function
$f(\pi)$	value of play π
$\mu \in \mathbb{Q}$	worst-case threshold
$\nu \in \mathbb{Q}$	expected value threshold
$\mathsf{Reach}_G(T)$	reachability objective with target set T
$\mathsf{Buchi}_G(T)$	Büchi objective with target set T
Parity_G	parity objective
$\mathsf{TP}(\rho)$	total-payoff of prefix ρ
$\underline{\mathsf{TP}}(\pi)/\overline{\mathsf{TP}}(\pi)$	infimum/supremum total-payoff of play π
$\mathsf{MP}(\rho)$	mean-payoff of prefix ρ
$\underline{\mathsf{MP}}(\pi)/\overline{\mathsf{MP}}(\pi)$	infimum/supremum mean-payoff of play π
$\mathsf{TS}_T(\pi)$	truncated sum of weights up to first visit of T
$\mathsf{EL}(\rho)$	energy level of prefix ρ
$\mathsf{TotalInf}_G(\mu)$	inf. total-payoff objective for threshold μ
$\mathsf{TotalSup}_G(\mu)$	sup. total-payoff objective for threshold μ
$\mathsf{MeanInf}_G(\mu)$	inf. mean-payoff objective for threshold μ
$\mathsf{MeanSup}_G(\mu)$	sup. mean-payoff objective for threshold μ
$\mathsf{ShortPath}_G(T, \mu)$	shortest path obj. for target T and threshold μ
$\mathsf{Energy}_G(v_0)$	energy objective for initial credit $v_0 \in \mathbb{N}$

Miscellaneous

$\emptyset$	empty set
proj_{A_i}	projection on set A_i

$\mathsf{Attr}_G^{\mathcal{P}_i,n}(A)$	attractor in n steps for $\mathcal{P}_i$ for states $A \subseteq S$
$\mathsf{Attr}_G^{\mathcal{P}_i}(A)$	attractor for $\mathcal{P}_i$ for states $A \subseteq S$
$\Box$	LTL globally
$\diamondsuit$	LTL eventually (or finally)
$\{0\}^k$	k-dimension zero vector

Even-parity self-covering trees

$T = (Q, R)$	epSCT
Q	set of nodes
$\varsigma, \vartheta, \chi, \nu, \xi \in Q$	nodes
$R \subset Q \times Q$	set of edges
$\Theta \colon Q \to S \times \mathbb{Z}^k$	labeling function
$\Theta(\varsigma) = \langle t, u \rangle$	node labeling
$\varsigma \rightsquigarrow \vartheta$	path from node ς to node ϑ
$\mathsf{Anc}(\varsigma)$	ancestors of node ς
$\mathsf{EnAnc}(\varsigma)$	energy ancestors of node ς
$\mathsf{oea}(\varsigma)$	oldest energy ancestor of node ς
d	branching degree of the game
l	depth of the tree
L	width of the tree
$D = (Q, R)$	directed acyclic graph (DAG)
$\mathsf{merge}(D)$	merge of DAG D

Synthesis in multi energy games

$\mathbb{C}$	bound on considered winning credits
$\mathsf{Cpre}_{\mathbb{C}}$	controllable predecessor operator
$\mathsf{Cpre}_{\mathbb{C}}^*$	fixed point of operator $\mathsf{Cpre}_{\mathbb{C}}$

Window games

$l_{\max}$	maximal window size
$\mathsf{GW}_G(v, l_{\max})$	good window objective (threshold v)
$\mathsf{DirFixWMP}_G(v, l_{\max})$	direct fixed window mean-payoff objective
$\mathsf{DirBndWMP}_G(v)$	direct bounded window mean-payoff objective

$\mathsf{FixWMP}_G(v, l_{\max})$	fixed window mean-payoff objective
$\mathsf{BndWMP}_G(v)$	bounded window mean-payoff objective