Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 59-10 (2025) 1384–1389

Scheduling of Pick-and-Place Tasks in Shared Workspaces Using a Trajectory-Free Approximation Method

Benjamin Oberthür * **Younes El Ghazi** ** **Arnaud Laurent** ** **Olivier Cardin** **

* *Mathematics and Operation Research Department, University of Mons, Belgium (email: benjamin.oberthur@umons.ac.be)*

** *CNRS, LS2N, École Centrale Nantes, Nantes University, France (e-mail: firstname.surname@univ-nantes.fr, with dash for compound surnames)*

Abstract: The scheduling of pick-and-place tasks in systems with multiple robots in shared workspaces is a well-studied topic with significant industrial relevance. We address a complex scenario involving four heterogeneous robot arms operating within a limited shared workspace, which was introduced by El Ghazi et al. (2024). Their approach leverages mixed-integer linear programming (MILP) modeling using trajectory simulations, achieving low makespans but at the cost of high computational time and model complexity.

To improve efficiency, we propose an alternative approximation method based only on the durations of the trajectories and the relative positions of robots and objects, without prior knowledge of the space taken by the robots in their trajectory, therefore bypassing the need for spatial trajectory simulation. Our model incorporates *hitboxes* to approximate the space occupied by robots during tasks, therefore predicting collisions between robots, while reducing computational load.

Experimental comparisons indicate that our method is significantly faster than the other approach. For larger instances (ten objects to pick up), our model often outperforms El Ghazi et al.'s model in terms of makespan.

Keywords: Scheduling, Pick-and-Place, Collision avoidance, MILP, Heuristic, Robotic arms

1. INTRODUCTION

Optimization of pick-and-place systems scheduling is a very well-studied subject (Alatartsev et al., 2015), with a high applicability in the industrial field. Different types of pick-and-place systems have been considered, from single-robot ones (Han et al., 2020), to multiple robots in a shared workspace (Åblad et al., 2018; Chen et al., 2022).

In this article, we discuss the problem presented in El Ghazi et al. (2024). This problem is particularly complex, as it considers a pick-and-place system composed of multiple heterogeneous robot arms in a shared workspace. We are given the position of a set of robots R and a set of objects O to pick up, and for all pairs (r, o) of robot/object, the fastest trajectory for which r picks up o and places it in its drop-off area. The objects are initially located in a shared workspace surrounded by the robots, in which all the robots work. With these data, the objective is to find a schedule specifying when each robot should pick up each object. This schedule should minimize the total time required for all the objects to be taken away from the workspace by a (single) robot and placed in the robot's drop-off area. The paths and their duration are computed using the robot operating system ROS (Takaya et al., 2016). However, certain objects may be too far from certain robots, adding a constraint to the scheduling problem. The main constraint of the problem is to return a solution where the robots do not collide with each other when they fetch objects (see in Figure 1 for a 2D visualization of the system).

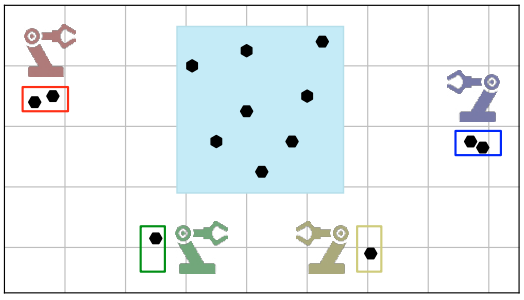

Fig. 1. Graphic representation of the system. The large cyan square represents the workspace. The colored rectangles represent the drop-off areas of the different robots. The black hexagons represent objects that the robots must pick.

The method proposed by El Ghazi et al. (2024) (which we refer to as El Ghazi's method in the remainder of this paper) uses MILP modeling to solve this problem, and returns low makespans, but takes a very long time to

10.1016/j.ifacol.2025.09.233

build the model and solve it. To address this limitation, we develop an approximation method to schedule the tasks based only on the positions of the robots and objects, as well as the duration of the different possible trajectories, but not the actual space occupied by the robots during their trajectories.

This article is organized as follows: Section 2 reviews the work on multi-robot task scheduling. Section 3 presents El Ghazi's method. Section 4 discusses the methods proposed to approximate the collisions between the robots without knowing the actual space taken by the robots during their trajectories. The scheduling model using the approximation method is presented in Section 5. We then compare those methods' performance in Section 6. And finally, we conclude and expose the possible future work on this subject in Section 7.

2. RELATED WORK

In Xin et al. (2020) and Kurosu et al. (2017), the pick-and-place problem is considered, with robots arms having to pick up objects and putting them at arbitrary positions. This problem seems close to ours, except that since the objects have to be put at arbitrary positions, the order in which a robot takes the objects not only depends on the position of the other robots, but should also take into account the distance between the ending positions of an objet and the starting position of the next one. The problem is therefore solved in the articles as a variant of the mTSP – a TSP with multiple salespeople.

In Hagebring et al. (2016); Aarts (2018) and Roselli (2022), different kinds of pick-and-place problems are considered, solved using MILP modeling. However, the robots considered are not robot arms, but stacker crane for the first article, and autonomous mobile robots (AMRs) for the two others. The stacker cranes from Hagebring et al. (2016) move along a single linear track, taking one discrete slot at a time and must work together to pick up and drop off packages on shelves adjacent to their current slots. And the two other, since the robots are AMRs, there is not the constraint of non-reachability of certain objects by some robots.

In Åblad et al. (2018), the authors tackle the multiple robot pick-and-place problem, where the whole workspace can be partitioned among the different robots – i.e. the workspace of each robot cannot overlap. We cannot make such assumption in our case, at the risk of getting a suboptimal solution.

In Chen et al. (2022), the authors address the multi-arm assembly planning problem, where several robot arms must manipulate objects to put them in specific configuration – such as (un)stacking toy bricks. Their approach is similar to El Ghazi's, with a more detailed trajectory simulation, and the problem, though including multiple robots, is not properly a pick-and-place problem.

In Kimmel and Bekris (2016), the authors propose methods to sort two groups of objects using a dual-arm system and place them into designated bins. While effective, this method requires substantial computation time for the coordination diagram, often exceeding one hour.

A connected subject to the one treated in this article is robot cell scheduling. A robot cell consists in an assembly line composed of multiple robots where the different resources are handled and distributed to the robots by a central robot arm. This problem is treated in Dawande et al. (2005) and Finke and Brauner (2000). However, robot cell scheduling does not seem to require collision avoidance management.

3. EL GHAZI ET AL.'S METHOD

We quickly recall the practical problem: given a set of robots R and a set of objects O, how to plan the different robots so that they fetch all the objects as fast as possible (i.e. minimize the makespan $C_{\max}$ of the system) without getting any collision between the robots? In this section, we first briefly present El Ghazi's method.

To solve this problem, the workspace is partitioned into q^3 *incidence cubes* (q per dimension) and the fastest trajectories for each pair robot/object are planned. The trajectories are then divided into steps of t milliseconds (ms), and simulated with the incidence cubes, so that we know which cubes the robot passes by at each step for each trajectory. Once those steps are computed, the goal is to assign them to t ms-long time slots – each robot is given a large enough number of those time slots – under certain constraints:

(1) One and only one of the trajectories corresponding to a given object must be executed (i.e. each object must be fetched by exactly one robot).

(2) All the steps of a trajectory must be executed in order (with potential pauses between those steps)

(3) Two steps executed at the same time (by different robots) must not use a common incidence cube, otherwise it means that the two robots collide (according to the simulations).

To avoid collisions, the robots have the possibility to stop in their course to let another robot pass (as indicated in point (2)). This problem is then modeled and solved with a MILP solver (Gurobi). As described in Section 6, this model returns solutions with a very accurate detection of collisions on our practical case. But as the number of variables and constraints depend on the number of objects, time slots, cubes and steps in the trajectories, the model can take a lot of time to be built and solved – often more than an hour.

In this paper, we propose a method that does not rely on spatial trajectory simulations, and where the robots cannot stop in their trajectory (only between two trips) in order to obtain solutions more quickly.

4. COLLISION AVOIDANCE

The first difference between our method and El Ghazi's one is that, for simplicity's sake, we work on two dimensions. In particular, the incidence cubes are projected on the (x, y) plan, and a trajectory is said to be incident to a square if it is incident to at least one of the q cubes of same (x, y) coordinates.

Since without trajectory simulation, we cannot know the space that a robot will occupy during its trajectories, we need to approximate it. For that, we consider a zone around the segment joining the robot and the object of a given trajectory, in which the robot may pass. We call it the *hitbox* of the trajectory. The robots of two trajectories (r_1, o_1) and (r_2, o_2) are said to collide if one of the following properties is verified:

(1) the segments joining r_1 to o_1 and r_2 to o_2 cross,
(2) the object o_2 is in the hitbox of (r_1, o_1),
(3) the object o_1 is in the hitbox of (r_2, o_2).

To help us in the determination of possible good hitbox shape, we provide a visualization of the space taken by some robots during some of their trajectories. The construction of two shapes of hitboxes, the parabola and the ellipse, are then presented.

4.1 Trajectory shape visualization

To find a good hitbox shape, we try and find if there is a certain recurring shape in the space that is taken by the robots in their different trajectories, using the simulations from El Ghazi's method. In Figure 2, we can see the squares incident to different trajectories from different instances. No pattern seems to emerge from these figures, as sometimes the robot makes a curve to the left as in subfigure (a), sometimes to the right as in subfigure (b), or goes straight to the object as in subfigure (c). Therefore, we need to consider an hitbox shape that covers widely around the segment connecting the robot to the object. For that matter, we consider two shapes for a hitbox: the parabola and the ellipse. We describe them in the two following subsections.

4.2 Parabolic hitbox

From now on, we denote the trajectory of the robot r to fetch the object o as (r, o). We define the parabolic hitbox as a tensor P such that $P^r_{oo'} = 1$ if the object o' is in the parabola covering (r, o), otherwise $P^r_{oo'} = 0$. To be able to decide if o' is in the parabola, we need to describe the latter. Let $\mathcal{P}^r_o$ be the Cartesian frame of origin p_o (the coordinates of the object o); with its y-axis $y_{\mathcal{P}^r_o}$ represented by the vector $\overrightarrow{p_o p_r}$, and its x-axis $x_{\mathcal{P}^r_o}$ being a rotation of angle $-\dfrac{\pi}{2}$ of the y-axis. In this new frame, we define the parabola describing the hitbox as the one passing the points $(-d_P, 0)$, $(0, -d_P)$, and $(d_P, 0)$, with d_P a size parameter of the parabola. Its equation is then $y = \dfrac{1}{d_P}x^2 - d_P$. So if the coordinates $p_{o'}(\mathcal{P}^r_o) = (x_{o'}(\mathcal{P}^r_o), y_{o'}(\mathcal{P}^r_o))$ of o' in $\mathcal{P}^r_o$ are such that

$$y_{o'}(\mathcal{P}^r_o) \geq \frac{1}{d_P}(x_{o'}(\mathcal{P}^r_o))^2 - d_P,$$

then o' is in the hitbox of (r, o), and $P^r_{oo'} = 1$.

Let $C^2 = (O, x_{C^2}, y_{C^2})$ be the usual Cartesian frame of the plane, and $p(C^2)$ the coordinates of the point p in the usual Cartesian frame C^2. To compute the coordinates $p(\mathcal{P}^r_o)$ of a point p in the new frame, we apply the following algorithm:

(1) **Computation of a vector collinear to $x_{\mathcal{P}^r_o}$ of same direction:** Let $u = (x_u, y_u) = \overrightarrow{p_o p_r}$ in C^2. The vector $v = (y_u, -x_u)$ is such that the directed angle $\widehat{\overrightarrow{u}, \overrightarrow{v}} = -\frac{\pi}{2}$, and is therefore collinear to $x_{\mathcal{P}^r_o}$ and shares the same direction.

(2) **Computation of θ:** We want to compute the directed angle $\theta = \widehat{\overrightarrow{x_{\mathcal{P}^r_o}}, \overrightarrow{x_{C^2}}}$ between the x-axis of the two frames. By definition, $x_{C^2} = (1, 0)$, and $v = k \cdot x_{\mathcal{P}^r_o}$ with $k > 0$. With the arccosine function, we can find the value of the angle:

$$|\theta| = \cos^{-1}\left(\frac{x_{\mathcal{P}^r_o} \cdot x_{C^2}}{\|x_{\mathcal{P}^r_o}\|\|x_{C^2}\|}\right) = \cos^{-1}\left(\frac{v \cdot x_{C^2}}{\|v\|\|x_{C^2}\|}\right) = \frac{x_v}{\|v\|}$$

The sign of θ is positive if $y_v \leq 0$, or equivalently $x_u \geq 0$, else it is negative.

(3) **Computation of $p(\mathcal{P}^r_o)$:** $p(\mathcal{P}^r_o)$ is the composition of the translation of vector $\overrightarrow{p_o(C^2)O}$ and the rotation of angle θ, applied to the point $p(C^2)$. Therefore its coordinates are defined by: $p(\mathcal{P}^r_o) = R_\theta\left(p(C^2) - p_o(C^2)\right)$, with the rotation matrix of angle θ:

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

The reader can see in Figure 3a an idea of the shape of the parabolic hitbox, with the representation of the different used notations. We determine the optimal size of the ellipse (i.e. the optimal value of d_P) in Section 6.

4.3 Elliptic hitbox

In the same way as for the parabolic hitbox, we define the elliptic hitbox as a tensor E such that $E^r_{oo'} = 1$ if the object o' is in the ellipse encircling (r, o). As in the previous subsection, we use a new frame to decide whether o' is in the hitbox of a trajectory. Let $\mathcal{E}^r_o$ be the Cartesian frame of origin the middle point between p_o and p_r, with its x-axis represented by the vector $\overrightarrow{p_r p_o}$, and its y-axis being a rotation of angle $\dfrac{\pi}{2}$ of the x-axis. We define the elliptic hitbox of (r, o) as the standard ellipse in $\mathcal{E}^r_o$ of focus points $p_r(\mathcal{E}^r_o)$ and $p_o(\mathcal{E}^r_o)$ with a distance d_E between the focus points and their closest vertex. So, with c the distance between the center of the ellipse and either of its focus points, $a := c + d_E$ the semi-major axis, and $b := \sqrt{a^2 - c^2}$ its semi-minor axis, the equation of the hitbox is $\dfrac{x^2}{a^2} + \dfrac{y^2}{b^2} = 1$. And finally, o' is in the hitbox of (r, o) if and only if :

$$\frac{(x_{o'}(\mathcal{E}^r_o))^2}{a^2} + \frac{(y_{o'}(\mathcal{E}^r_o))^2}{b^2} \leq 1.$$

The computations to pass from the usual frame to $\mathcal{E}^r_o$ are similar to the ones for $\mathcal{P}^r_o$ and are not presented here.

The reader can see in Figure 3b an idea of the shape of the elliptic hitbox with the different used notations. As for the parabolic hitbox, we determine the optimal value for d_E in Section 6 and compare which hitbox is better. For now, we use a general hitbox notation H in our model, such that $H^r_{oo'} = 1$ if o' is in the hitbox covering (r, o).

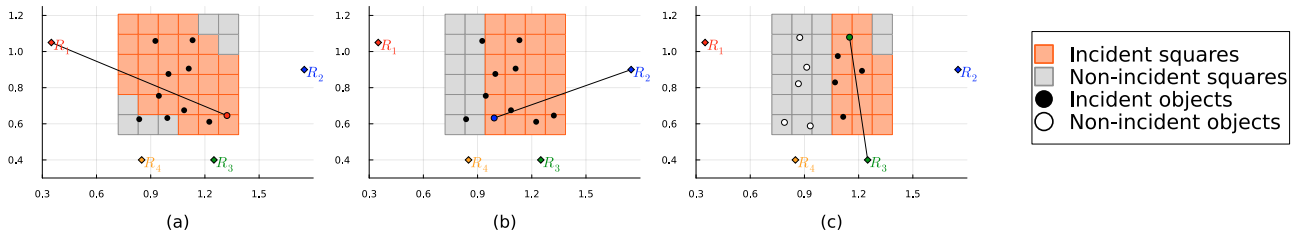

Fig. 2. Intersections of the trajectories of the colored robots to their corresponding colored object.

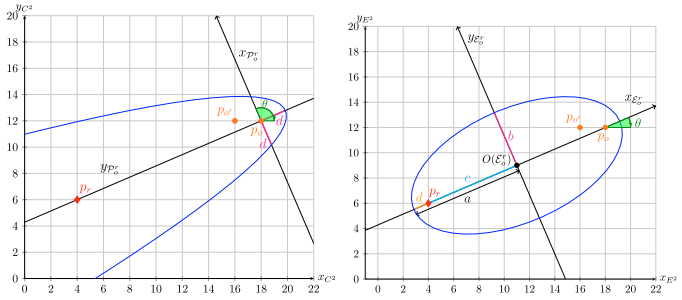

Fig. 3. Examples of the parabolic and elliptic hitboxes encircling a trajectory

5. SCHEDULING MODEL

Now that we have an approximation of the different trajectories, we can now schedule the robots to make them fetch the objects. We model this problem as an **unrelated-machine single-stage job scheduling problem**. The problem is single-stage as each object corresponds to a single task. The machines (in our case, the robots) are unrelated as the fetching times of a given robot are not proportional to the fetching times of the others. We call the following model M_S. The data, variables, objective function and constraints are presented hereunder.

Data:

O**:** Set of indices of the objects

R**:** Set of indices of the robots

$t_o^r \in \mathbb{R}_+ \cup \{+\infty\}$**:** Time taken by the robot $r \in R$ to pick up the object $o \in O$. If the robot r cannot reach the object o, then $t_o^r = +\infty$.

$C_{oo'}^{rr'} = 1$ if the line segments $[p_o, p_r]$ and $[p_{o'}, p_{r'}]$ intersect, 0 else.

$H_{oo'}^r = 1$ if the point $p_{o'}$ is in the hitbox covering (r, o).

We use the tensor C so that, if two trajectories cross each other, even though neither object is in the hitbox of the other trajectory, we still detect the collision.

Variables:

$t_o \in \mathbb{R}_+$**:** Starting time of the job of fetching the object $o \in O$.

$x_o^r \in \{0, 1\}$**:** $= 1$ if the object $o \in O$ is assigned to the robot $r \in R$. If r cannot reach o, we fix x_o^r to 0.

$a_{oo'} \in \{0, 1\}$**:** $= 1$ if the task of fetching the object o is done before the task of fetching the object $o' \neq o$ starts.

$C_{\max} \in \mathbb{R}_+$**:** Makespan of the whole execution of the system

Model:

$$\min C_{\max} \tag{1}$$

subject to

$$C_{\max} \geq t_o + \sum_{r \in R} t_o^r \cdot x_o^r, \qquad \forall o \in O \tag{2}$$

$$\sum_{r \in R} x_o^r = 1, \qquad \forall o \in O \tag{3}$$

$$t_o + \sum_{r \in R} t_o^r \cdot x_o^r \leq t_{o'} + M(1 - a_{oo'}), \forall o, o' \in O;\ o \neq o' \tag{4}$$

$$a_{oo'} + a_{o'o} \leq 1, \qquad \forall o, o' \in O;\ o \neq o' \tag{5}$$

$$a_{oo'} + a_{o'o} \geq x_o^r + x_{o'}^r - 1, \quad \forall r \in R, \forall o, o' \in O;\ o \neq o' \tag{6}$$

$$x_o^r + x_{o'}^{r'} - a_{oo'} - a_{o'o} \leq 1, \quad \forall r, r' \in R;\ r < r', \forall o, o' \in O;\ o \neq o' \wedge C_{oo'}^{rr'} = 1 \tag{7}$$

$$x_o^r - x_{o'}^r - a_{oo'} - a_{o'o} \leq 0, \quad \forall r \in R, \forall o, o' \in O;\ o \neq o' \wedge H_{oo'}^r = 1 \tag{8}$$

Constraint (2) defines the makespan variable $C_{\max}$ that we minimize in the objective function (1). Constraint (3) forces each object to be assigned to exactly one robot. Constraints (4) to (6) define the precedence variable a: constraint (4) means that if $a_{oo'} = 1$ then the task of picking up o' must start after the task of picking up o has ended, otherwise the variable should be set to 0; constraint (5) models the fact that $a_{oo'}$ and $a_{oo'}$ cannot both be set to 1; and constraint (6) asks that if o and o' are both picked up by the same robot r, then one of the two variables must be set to 1. Constraint (7) prevents o and o' to be respectively fetched at the same time by r and r' if the segments $[p_o, p_r]$ and $[p_{o'}, p_{r'}]$ intersect. And finally constraint (8) means that if o' is in the hitbox covering (r, o), then object o' cannot be fetched by any robot while object o is fetched by robot r.

In this model, the space covered by the hitbox of (r, o) is reserved for all the duration of the task of r fetching o. During this time, only objects o' not in (r, o)'s hitbox can be fetched by another robot r', as long as the segments joining r to o and r' to o' do not intersect and o is not in (r', o')'s hitbox. This adds a constraint compared to

El Ghazi's model where the robots could stop in their trajectories to let another robot pass, and continue their course when the path was clear. So we can expect our objective values to be larger than the other model, but with better computation times.

6. RESULTS

In this section, we present the experimental environment in which we worked, identify the best hitbox to use in our model, and finally compare the performance of the two discussed models.

6.1 Instances and computation environment

The results obtained in this section are computed with Gurobi version 10.0.3 on an 11th Gen Intel® Core™ i7-1185G7 CPU @ 3.00GHz × 4 with 32 GB of RAM. The instances used to compare the two models contain $n = 4$ robots and $m = 4$, 6, 8, or 10 objects. The workspace in which the objects are put is a square of side $\Delta = 0.55$cm or 0.35cm (always centered on the same point). In El Ghazi's model (M_{EG}), the workspace is divided in $q = 6$ cubes of equal size per dimension, and the time and trajectories of the robots are respectively divided into slots and steps of $t = 100$ms. For each pair (m, Δ), there are 10 different instances. We define the instances with $\Delta = 0.35$ as *tight*, and the ones with $\Delta = 0.55$ as *sparse*. For each instances, we stop the resolution after one hour, but we impose no limit on the construction time of the models.

6.2 Hitbox performance comparison

We determine for which values of d_P and d_E the two hitboxes perform the best. For that, we compute the F1-score of the hitboxes for different values of the parameters. The F1-score is defined as the harmonic mean of the precision (the proportion of actual collisions among those detected) and the recall (the proportion of predicted collision among the actual ones) of a simulation. This score can be expressed in function of the number of true positives (TP), false positives (FP), and false negatives (FN):

$$F1 = \frac{2}{\text{precision}^{-1} + \text{recall}^{-1}} = \frac{2TP}{2TP + FP + FN}.$$

In our case, we define the number of TP as the number of pairs of trajectories that are detected as colliding during at least one time step in El Ghazi's simulations (our reference), and for which the segments (r, o) and (r', o') cross, and/or one object of the two trajectories is in the hitbox of the other. In the same idea, the number of FN is defined as the number of cases where a collision is detected in the El Ghazi's simulations, but not in ours. Therefore, a high F1-score represents a model that predicts well the collisions in a large proportion. As the hitboxes' sizes depend on their respective parameter d_P or d_E, so do TP, FP and FN, and thus the F1-score. We compute this score for each instances provided by El Ghazi, and we find a value of d_P and d_E that seems to maximize the mean F1-score over all the instances.

In Figure 4, we can see the mean F1-score of the parabolic hitbox for some values of d_P, and of the elliptic hitbox for some values of d_E, respectively denoted $mean(F1_P(d_P))$ and $mean(F1_E(d_E))$. This shows us that the functions seem to each have a global optimum. We look for the value of d_P and d_E in a range of step 0.001 that maximizes their respective function and we get $d_P^* = 0.232$ and $d_E^* = 0.162$. Table 1 gathers d_P^* and d_E^* along with their associated mean F1-score, precision and recall. The ellipse of size 0.162 has a better F1-score and recall, but a bit lower precision than the parabola of size 0.232. We choose the elliptic hitbox as it leaves less undetected collision than the parabola. Even though some collisions detected by El Ghazi's simulation are not detected by our elliptic hitbox (the recall being less than 1), since the partition of the workspace proposed by El Ghazi is not so precise, some collisions found with their simulation may not be actually happening. So, the recalls computed may overall be higher.

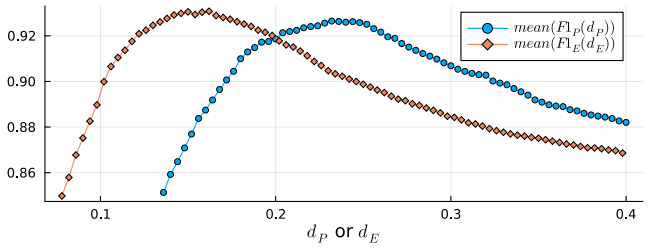

Fig. 4. Plot of the mean F1-score of the parabolic and elliptic hitboxes over the different instances, for $d_P \in [0.136, 0.4]$ and $d_E \in [0.074, 0.4]$, with a step of 0.004.

Table 1. Best sizes of the two hitbox shapes and their respective average scores

	d^*	F1	Precision	Recall
Parabola	0.232	0.926	0.920	0.934
Ellipse	0.162	0.931	0.914	0.949

6.3 Comparison of the models

With the chosen hitbox, we now compare our models. Let z_S^* and z_{EG}^* be the best solutions found by Gurobi respectively with M_S and M_{EG} (as a reminder, respectively our scheduling model, and El Ghazi's model) on a given instance. We define the Relative Percentage Deviation (RPD) of z_S^* compared to z_{EG}^* as $\text{RPD} = \frac{z_{EG}^* - z_S^*}{z_{EG}^*}$.

In Table 2, we can see the construction time of the model added to its solving time by Gurobi (the two distinct values were not available in El Ghazi's data), the RPD between the two models, and the duality gap of the solution returned by Gurobi on M_{EG} (i.e., the relative deviation between the last primal and dual solutions found). All the values are averaged over the 10 instances of same type (same number of objects and size of workspace). These results show that, as expected, our model is faster than El Ghazi's, as it is never constructed and solved in more than 0.4 second. However, we can see that for instances up to 8 objects, our solutions are worse than those returned with M_{EG}. But on the other hand, the duality gap of the solution returned with M_{EG} rises as the instances get larger, meaning that Gurobi's solution are most likely far from their optimum. Therefore, within one hour of solving, our model returns solutions with lower makespans than El Ghazi's for instances with 10 objects.

Table 2. Comparative performance results of the two models. (ARPD = Average RPD)

		Construction + solving time (s)		ARPD (%)	M_{EG}'s duality gap
		M_{EG}	M_S		
$m = 4$	tight	785.74	0.032	-59.07	0.0
	sparse	1975.78	0.0071	-29.79	7.05
$m = 6$	tight	4870.52	0.014	-68.25	39.08
	sparse	4820.42	0.013	-26.35	46.79
$m = 8$	tight	6009.80	0.026	-15.85	66.06
	sparse	6445.20	0.048	-9.38	65.88
$m = 10$	tight [1]	6617.77	0.11	**16.01**	80.17
	sparse	8004.37	0.12	**17.00**	77.38

7. CONCLUSION

In this article, we exposed a novel difficult pick-and-place problem with multiple robots in a shared limited workspace introduced in El Ghazi et al. (2024). The method presented in the latter, which we used as a point of comparison, is based on trajectory simulations in a partition of the space, and a MILP model whose number of variables and constraints depends on the precision of the space partitioning and time discretization (among other quantities). Our method only uses the time of the trajectories of the robots, and the respective positions of the robots and objects. The room taken by the robots in their different trajectories is approximated using a hitbox whose size depends on the positions of the robot and the object of the given trajectory. Two shapes of hitbox were considered, the parabola and the ellipse. After running tests using the F1-score measure to compare the quality of an hitbox of given size, a certain size of ellipse stood out as a well performing hitbox for predicting collisions. We used this hitbox in a scheduling model to compare both model's performances. Results showed that our model is substantially faster than El Ghazi's, and that, due to the size of El Ghazi's model, the solutions found by Gurobi after one hour of solving get farther from the optimal as the instances get bigger. Therefore, for instances with 10 objects, despite the fact that our model is an approximation of El Ghazi's model, we get smaller makespans with our model than with theirs. However, we need to keep in mind that our hitbox does not detect all collisions detected by El Ghazi's simulations, even if the large size of the cubes used in the partition of the workspace for their simulations are likely to lead to the detection of more collisions than there actually are.

These results, however promising for large instances, are still to be put in perspective. The simulations of El Ghazi being on their own an approximation of the reality (with their partition of the space with cubes), our approximation (the hitboxes) is therefore a farther approximation of the reality. A solution to cope with this problem would be to redo the El Ghazi's simulations with a finer partitioning. Also, the schedules returned by our model could be tested with El Ghazi's simulations to check the percentage of schedules that contain collisions detected by El Ghazi's.

[1] As Gurobi could not find a solution with M_{EG} after 1 hour on one of the 10 tight instance of 10 objects, we compute the average values over the 9 other instances.

REFERENCES

Aarts, J. (2018). *Simultaneous Multi-Robot Task Scheduling and Path Planning*. Master's thesis, TU Delft.

Alatartsev, S., Stellmacher, S., and Ortmeier, F. (2015). Robotic task sequencing problem: A survey. *Journal of intelligent & robotic systems*, 80, 279–298.

Chen, J., Li, J., Huang, Y., Garrett, C., Sun, D., Fan, C., Hofmann, A., Mueller, C., Koenig, S., and Williams, B.C. (2022). Cooperative task and motion planning for multi-arm assembly systems.

Dawande, M., Geismar, H.N., Sethi, S.P., and Sriskandarajah, C. (2005). Sequencing and scheduling in robotic cells: Recent developments. *Journal of Scheduling*, 8, 387–426.

El Ghazi, Y., Cardin, O., Laurent, A., Subrin, K., Levilly, S., and Mouchère, H. (2024). Optimizing Task Allocation and Movement Coordination in Autonomous Robotic Systems. *SSRN*. URL `https://ssrn.com/abstract=4964992`.

Finke, G. and Brauner, N. (2000). Scheduling in robotic cells. In *Operations Research Proceedings 1999*, 372–377. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hagebring, F., Wigström, O., Lennartson, B., Ware, S.I., and Su, R. (2016). Comparing milp, cp, and a* for multiple stacker crane scheduling. In *2016 13th International Workshop on Discrete Event Systems (WODES)*, 63–70. doi:10.1109/WODES.2016.7497827.

Han, S.D., Feng, S.W., and Yu, J. (2020). Toward fast and optimal robotic pick-and-place on a moving conveyor. *IEEE Robotics and Automation Letters*, 5(2), 446–453. doi:10.1109/LRA.2019.2961605.

Kimmel, A. and Bekris, K.E. (2016). Scheduling pick-and-place tasks for dual-arm manipulators using incremental search on coordination diagrams. In *ICAPS workshop on planning and robotics (PlanRob)*.

Kurosu, J., Yorozu, A., and Takahashi, M. (2017). Simultaneous dual-arm motion planning for minimizing operation time. *Applied Sciences*, 7(12).

Åblad, E., Spensieri, D., Bohlin, R., and Carlson, J.S. (2018). Intersection-free geometrical partitioning of multirobot stations for cycle time optimization. *IEEE Transactions on Automation Science and Engineering*, 15(2), 842–851. doi:10.1109/TASE.2017.2761180.

Roselli, S.F. (2022). *Conflict-Free Routing of Mobile Robots*. Ph.D. thesis, Chalmers (Sweden).

Takaya, K., Asai, T., Kroumov, V., and Smarandache, F. (2016). Simulation environment for mobile robots testing using ros and gazebo. In *20th International Conference on System Theory, Control and Computing (ICSTCC)*, 96–101. doi:10.1109/ICSTCC.2016.7790647.

Xin, J., Meng, C., Schulte, F., Peng, J., Liu, Y., and Negenborn, R.R. (2020). A time-space network model for collision-free routing of planar motions in a multirobot station. *IEEE Transactions on Industrial Informatics*, 16(10), 6413–6422. doi:10.1109/TII.2020.2968099.