A Dynamic Programming Approach for Cooperative Pallet-Loading Manipulators

A Dynamic Programming Approach for Cooperative Pallet-Loading Manipulators

Luca Consolini, Mattia Laurini and Marco Locatelli

***Abstract*—In a high-speed palletizing machine, packages of various sizes are inserted on a conveyor belt. Then, cooperating multiple robotic manipulators move them to obtain a desired final layout. The throughput of this palletizing process critically hinges upon the strategic selection of the insertion sequence and the careful choice of robot manipulations. Pursuing a higher throughput in this context holds great importance due to its potential to enhance productivity, however, reaching such goal constitutes a challenging task. Indeed, the problem of maximizing the throughput of the palletizing machine is a nontrivial one and, despite its relevant importance in industrial settings, it has not received much attention in existing literature. In this work, we present a Dynamic Programming-based algorithm, together with some reduction techniques, that allows finding the shortest packages sequence and the corresponding robot manipulations that maximize production. We include some numerical experiments on randomly generated problems and on actual industrial scenarios, which show the good performance of the proposed method.**

***Note to Practitioners*—This work is motivated by the need of high-speed palletizing machine manufacturers to automate the generation of packages sequences, and the corresponding robot manipulations tasks assignment. We solve this problem with a Dynamic Programming-based algorithm. The benefit of the proposed method is twofold. On one hand, it allows palletizing machines manufacturers not to waste their employees' time on the often lengthy task of manually planning packages sequences and manipulations. On the other hand, the proposed approach allows minimizing the time required to assemble an assigned layout, increasing the overall throughput of the production chain. The proposed algorithm can be implemented in any programming language of choice (e.g., C++) and integrated by manufacturers in their production software. The main limitation of this approach is the computational time which grows exponentially with the number of packages. However, given that the application is an off-line one, this approach allows handling most of the industrial layouts, which usually consist of a few tens of packages, in a reasonable amount of time. As future developments, the approach could be generalized to handle more complicated manipulator movements and/or allow robots to manipulate each package more than once. This would add a layer of complexity that would require nontrivial tailored solution strategies in order to handle these new degrees of freedom.**

***Index Terms*—Cooperative manipulators, manipulation planning, collision avoidance, dynamic programming.**

I. Introduction

We consider a high-speed palletizing machine with multiple robotic arms (see Figure 1). We fix an inertial coordinate frame along the moving belt. Assuming that the conveyor belt moves from left to right, the origin is placed at the position that the bottom-left corner of the belt occupies at the initial time. We assume that the x-axis is parallel to belt velocity.

A multiple-line infeed conveyor inserts packages of different sizes on the belt, that moves with constant speed. Figure 2 shows a possible packages configuration after their insertion on the belt, along two possible insertion lines, associated to different y-coordinates. Multiple robotic manipulators reposition packages along the y-axis, so that all of them reach the y-coordinate and orientation that corresponds to an assigned pallet layout. Figure 3 shows a possible configuration obtained after the manipulations. A stopping bar at the end of the conveyor belt, orthogonal to it, aligns packages along the x-axis, so that they make a layer of a given pallet configuration (see the right end of Figure 1). The machine obtains the desired final layout by moving packages along the y-axis and aligning them with the stopping bar. This procedure is somehow reminiscent of computer game "Tetris", in which the player moves falling pieces sideways, while these fall to the bottom of the screen to compose a desired configuration. Finally, the obtained packages layout is transferred from the conveyor belt to the top of a pallet, constituting one of its layers.

Since the palletizing process can be the bottleneck of the whole packaging line (see [1], [2], [3], [4]), it is important to reduce the time needed to compose the desired layout. For instance, [3] observes that "*Factories have many steps in a production cycle and a step might be a bottleneck causing low productivity. Palletizing tasks are often required at each step to convey the product to the next step. Therefore, there are many palletizing tasks in a factory and these tasks are crucial within it. From the review of the production site, we came to the conclusion that it is possible to reduce overall working hours by reducing the time needed for palletizing*". Indeed, product packages typically accumulate upstream of the palletizing machine and the ability of the palletizer to produce fully loaded pallets heavily impacts the throughput of the production plant. For a fixed conveyor belt speed, the problem of maximizing the throughput is equivalent to that of minimizing the overall length of the packages sequence with respect to the conveyor belt reference frame (see Figure 1). In principle, to reduce the length of the packages sequence we could place the packages on the conveyor belt as tight as possible. However, since manipulators bases are fixed and the belt is moving, the robots can manipulate each package only in a limited time-window. Hence, with such a choice, the manipulators could not be able to move all packages in time to the desired final positions. Consequently, we need to add

All authors are with the Department of Engineering and Architecture of the University of Parma, Parco Area delle Scienze, 181/A, 43124 Parma, Italy. {mattia.laurini, luca.consolini, marco.locatelli}@unipr.it. This research has financially been supported by Regione Emilia-Romagna in project "ROBOT-A" of framework "POR FSE 2014/2020 Obiettivo Tematico 10", by OCME S.R.L. and by the Programme "FIL-Quota Incentivante" of University of Parma and co-sponsored by Fondazione Cariparma.

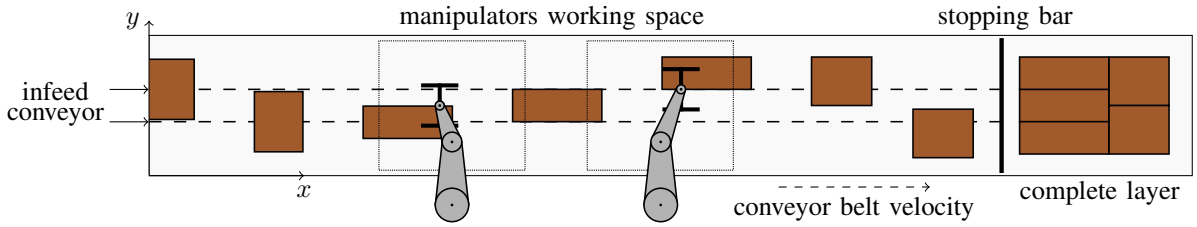


Fig. 1: A palletizing system with two manipulators, a stopping bar and a 2-line infeed conveyor.

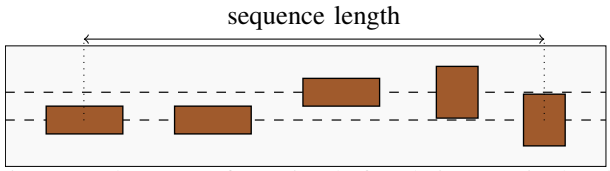


Fig. 2: Packages configuration before being manipulated.

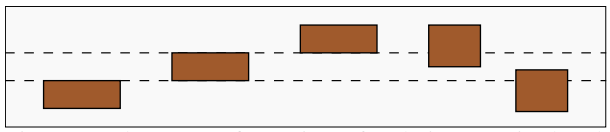


Fig. 3: Packages configuration after being manipulated.

sufficient spacing between packages, to give the robots enough time to execute all manipulations. Moreover, we also need to avoid collisions among packages during manipulations.

In a previous work of ours [5], we called this problem the Pallet Pattern Placement Sequence Problem (3PSP).

A. Literature review

The problem of optimizing the positions of some objects in a pallet, to reduce the occupied space, has been extensively studied in literature. Various works are related to the Bin Packing Problem (BPP) (see, e.g., [6], [7], [8]). Some of them address the problem in 2 dimensions, such as [9], [10], and with packages of different size, as in [11]. Other works consider the problem in 3 dimensions, like [12], or focus on the stability of stacked layers of products, as [13]. Some other works are devoted to the related Pallet Loading Problem (PLP) (see, e.g., [14], [15], [16]). It can be addressed in 2 dimensions, considering identical packages as in [17], and with secondary objectives, like in [18]. The PLP can also be addressed in 3 dimensions and with non-homogeneous packages (see, for instance, [19]).

The BPP and the PLP can be considered complementary to the 3PSP addressed in the present paper. Indeed, they focus on optimizing the final pallet layout, while we assume it to be known a priori. This assumption is common in industrial applications, since customers often provide their own pallet layouts to palletizing machines producers. Our goal is not that of maximizing the amount of packages on a pallet, as in PLP. Rather, we want to find the packages positions and the sequence of manipulations that allow to obtain an assigned layout in minimum-time. 3PSP takes into account manipulators kinematic and dynamic constraints, together with other factors, such as the conveyor belt speed, the robots working space, and the time windows in which every package can be manipulated by the robots. Moreover, it must avoid collisions among packages.

3PSP has not received much attention in literature. Some works consider it for high-speed palletizing machines (see [20]), or for different palletizing machines setups (see, e.g., [21], [22], [23], [24]). However, they describe the problem in very general terms, and do not provide a mathematical model or specific solution algorithms.

To the best of our knowledge, the only work in literature that provides a mathematical formulation of 3PSP is a previous work of ours [5]. There, we showed that 3PSP can be modeled as a Mixed Integer Linear Programming (MILP) problem. More specifically, we modeled it as a variant of a Vehicle Routing problem with time windows and additional collision avoidance constraints. In principle, such MILP model could be solved by standard solvers, such as GUROBI [25] or CPLEX [26]. However, the required computational time is very large. In order to reduce it, in [5], we proposed a suboptimal solution algorithm. However, such algorithm still relies on the use of commercial solvers, and the solution times are still very high for large problems. Moreover, the reduction of the problem to the MILP framework requires some simplifications, for instance in the formulation of the manipulators' dynamics.

Finally, note that the 3PSP can be considered a problem of cooperative robotics (see, e.g., [27], [28], [29], [30], [31]). In fact, in a good quality solution, manipulators carry out their tasks in a cooperative way, fulfilling the common goal of forming the prescribed layer with a minimum-length sequence.

B. Paper contribution

The main novelty of this paper is the use of Dynamic Programming (DP) for solving the 3PSP. The new approach has the following advantages:

- It is computationally efficient, since we developed two tailored cut strategies that allow improving computational times. Moreover, it does not need commercial solvers for solving 3PSP instances, making it more advantageous for companies that will not need purchasing an expensive license. In the numerical experiments section, we show the performance improvement due to the proposed cut strategies. Also, we will show that the proposed method outperforms the one introduced in [5], which makes use of the commercial solver GUROBI.
- It is flexible and allows handling complex constraints. For instance, we consider the presence of packages of different sizes and a multiple-line infeed conveyor. That is, with reference to Figure 1, we assume that packages can be inserted on the conveyor belt on different lines, corresponding to different y-coordinates. DP can handle manipulators with any dynamics, since manipulators travel

times can be represented by a generic function of initial and final positions.

C. Basic solution strategy

Let N be the number of packages that compose the desired final layout, M the number of manipulators, and K the number of infeed lines. We can represent a candidate manipulation sequence with the following decision variables, for $i \in \{1, \dots, N\}$,

- $x_i \in \mathbb{R}$, where x_i is the x-coordinate that the center of package i occupies when it is inserted on the conveyor belt;
- $y_i \in \{\hat{y}_1, \dots, \hat{y}_K\}$, where y_i is the y-coordinate that the center of package i occupies on the conveyor belt, associated to the chosen infeed line, and $\hat{y}_k$ is the y-coordinate associated to the k-th infeed line, with $k \in \{1, \dots, K\}$;
- $t_i \in \mathbb{R}, m_i \in \{1, \dots, M\}$, where t_i is the time at which package i is moved and m_i is the manipulator that moves it. Variables t_i and m_i completely describe the manipulation of the i-th package, indeed, as we will specify at point 4) of Section II-A, packages are only moved along the y-axis and the final y-coordinate of the i-th package, as well as its orientation, is known, as it depends only on the chosen layout.

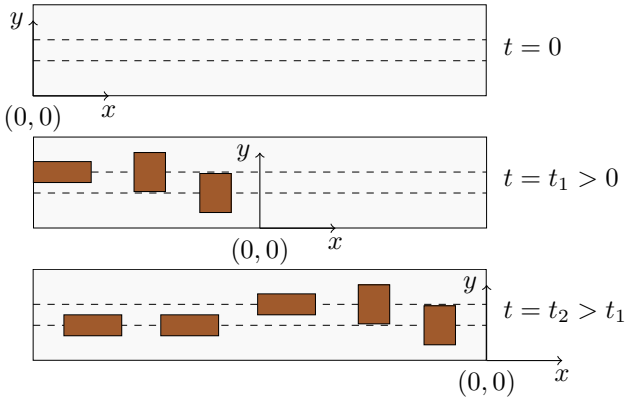


Fig. 4: Reference frame movement.

As said, the reference frame is fixed on the belt and the origin of the x-axis corresponds to the bottom-left corner of the conveyor belt at the initial time (see the top subfigure of Figure 4). Thus, coordinates x_i are nonpositive. Indeed, as seen in Figure 4, at initial time $t = 0$, the origin $(0, 0)$ occupies the bottom-left corner of the belt. At time $t_1 > 0$, the origin has moved to the right with respect to the world inertial frame and a few packages have been inserted on the belt with negative x-coordinates. Finally, at time $t_2 > t_1$, the origin has moved to the far right end of the conveyor belt and all packages have been placed on the belt with negative x-coordinates. Note that at times greater than t_2 the origin is positioned further right with respect to the end of the physical belt. So, packages x-coordinates are always nonpositive and if a package at position j is placed on the belt later than a package at position i, then $x_j < x_i$. We want to minimize the length of the packages sequence, or, equivalently, maximize the minimum among all variables x_i.

The resulting problem is a mixed integer non-linear one, having both continuous (x_i, t_i) and discrete (y_i, m_i) variables and involving non-linear constraints.

In our work [5], we formulated the 3PSP as a MILP. Here, we propose a DP approach that has lower computational times and can handle larger problems.

To limit the number of continuous decision variables, we discretize the set of packages insertion position x-coordinates. Namely, we assume that, for $i \in \{1, \dots, N\}$, $x_i \in \hat{\mathcal{X}}$, where $\hat{\mathcal{X}} = \{\hat{x}_1, \dots, \hat{x}_Q\}$ is a small cardinality set of available x-coordinates for inserting a package on the belt. Then, we iteratively optimize y_i, t_i, m_i, while keeping variables x_i constant, and optimize variables x_i, while keeping variables y_i, t_i, m_i constant. We will show that this strategy allows finding feasible solutions with low computational times. The obtained solutions are suboptimal, but in general of good quality. In fact, in Section VII-A, we will show that the aforementioned solutions cannot be improved by either modifying y_i, t_i, m_i, while keeping x_i fixed, or by modifying x_i along some directions, while keeping the other variables fixed.

II. Problem Modelization

A. Overall problem assumptions

To simplify the mathematical formulation of this problem, following [5], we make a number of assumptions, which hold true for many palletizing machines:

1) Manipulators operating spaces do not overlap.
2) Each package is moved only once.
3) At the end of the conveyor belt, there is a stopping bar orthogonal to it, which allows aligning packages.
4) With respect to the moving belt reference frame, packages are manipulated only along the y-axis. Moreover, note that manipulators can also perform $90°$ rotations of the packages according to their desired orientation in the final layout.
5) Packages are not lifted from the belt during manipulations. This is a common practice in industrial palletizing machines to improve the execution speed. Hence, we must make sure that packages do not collide with each other while in motion.
6) The conveyor belt speed w is constant and is considered a fixed parameter.

B. Associated graph

Similarly to [5], we define a graph $\mathbb{G}$, whose nodes are associated to initial and final packages and manipulators positions and whose edges represent the allowed transitions. Namely, we set $\mathbb{G} = (\mathcal{V}, \mathcal{E})$, in which the node set is $\mathcal{V} = \mathcal{O} \cup \mathcal{I} \cup \mathcal{F} \cup \mathcal{R}$. Set $\mathcal{O} = \{\mathcal{O}_1, \dots, \mathcal{O}_M\}$ represents the initial resting positions of the M manipulators, nodes in $\mathcal{I} = \{\mathcal{I}_1, \dots, \mathcal{I}_I\}$, with $I \geq N$, are associated to the initial available positions on the belt, $\mathcal{F} = \{\mathcal{F}_1, \dots, \mathcal{F}_N\}$ refers to packages final positions and $\mathcal{R} = \{\mathcal{R}_1, \dots, \mathcal{R}_M\}$ represents manipulators final resting positions. Since we assume to have a discrete set of possible packages position x-coordinates $\hat{\mathcal{X}} = \{\hat{x}_1, \dots, \hat{x}_Q\}$ and K infeed lines, the set of initial positions $\mathcal{I}$ has cardinality $|\mathcal{I}| = I = QK$. We set $\mathcal{N} = \mathcal{I} \cup \mathcal{F}$.

The edge set $\mathcal{E}$ is defined as follows:

- All nodes in $\mathcal{O}$ are connected to all nodes in $\mathcal{I}$, that is, manipulators can move from their initial resting position to an initial position on the belt occupied by a package.
- All nodes in $\mathcal{I}$ are connected to all nodes in $\mathcal{F}$. These edges represent the motions in which manipulators move a package from an initial to a final position.
- All nodes in $\mathcal{F}$ are connected to all nodes in $\mathcal{I}$. These edges represent the motions with which, after having moved a package, manipulators return to an initial package position to start another manipulation.
- All nodes in $\mathcal{F}$ are also connected to all nodes in $\mathcal{R}$. These edges represent those motions in which, after having moved a package, a manipulator goes to its final resting position.
- Each node in $\mathcal{O}$ is also connected to a corresponding node in $\mathcal{R}$. These edges represent the cases in which a manipulator $m \in \{1, \dots, M\}$ does not move any package, so that it goes directly from its initial resting position $\mathcal{O}_m$ to its final one $\mathcal{R}_m$.

Note that if a manipulator travels an edge in the last group (i.e., from $\mathcal{O}$ to $\mathcal{R}$), then it does not perform any manipulation, it is useless to the palletizing task and could be removed from the plant. Hence, edges of the last group should not be travelled in any appropriate solution. Anyway, we maintain these edges to represent solutions that do not use all manipulators.

The subgraph given by $(\mathcal{N}, \{e \in \mathcal{E} \mid e \in \mathcal{N} \times \mathcal{N}\})$ is bipartite, since manipulators are only allowed to move from an initial package position to a final one, (i.e., they perform a manipulation), and from a final package position to an initial one, (i.e., after having performed a manipulation, they move to a new initial package position to perform a new manipulation).

The adjacency matrix of graph $\mathbb{G}$ is

$$A = \begin{pmatrix} 0_{M,M} & 1_{M,I} & 0_{M,N} & I_{M,M} \\ 0_{I,M} & 0_{I,I} & 1_{I,N} & 0_{I,M} \\ 0_{N,M} & 1_{N,I} & 0_{N,N} & 1_{N,M} \\ 0_{M,M} & 0_{M,I} & 0_{M,N} & 0_{M,M} \end{pmatrix}, \tag{1}$$

with $A \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ and where, for any $\ell, m \in \mathbb{N}$, $0_{\ell,m}$, $1_{\ell,m}$ and $I_{\ell,\ell}$ are the $\ell \times m$ zero matrix, the $\ell \times m$ matrix with all entries equal to 1 and the $\ell \times \ell$ identity matrix, respectively.

At time $t = 0$, all manipulators occupy their initial position in $\mathcal{O}$ and, at the end, each manipulator is required to be at its final resting position in $\mathcal{R}$. The sequence of movements of the m-th manipulator describes a path on the graph that starts at $\mathcal{O}_m$. Then, the manipulator has two options: it may go directly to the corresponding final resting position $\mathcal{R}_m$ (in this case, the manipulator does not move any package and it travels along the edge represented by the m-th diagonal element of $I_{M,M}$ in (1)). Otherwise, the manipulator travels to a node in $\mathcal{I}$, then travels alternately between nodes in $\mathcal{F}$ and $\mathcal{I}$, and then, after having visited one last node in $\mathcal{F}$, it goes to its final resting position $\mathcal{R}_m$.

As a clarifying example, consider a layout with $N = 4$ packages as depicted in Figure 5, for a system with $M = 2$ manipulators, a single-line infeed conveyor and a discrete set of packages position x-coordinates $\hat{\mathcal{X}} = \{\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4\}$. The associated graph $\mathbb{G} = (\mathcal{V}, \mathcal{E})$, with node set $\mathcal{V} = \{\mathcal{O}_1, \mathcal{O}_2, \mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4, \mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4, \mathcal{R}_1, \mathcal{R}_2\}$, is depicted in Figure 6. In this example we consider a single-line infeed conveyor, thus $I = N = 4$. The manipulator at initial resting position $\mathcal{O}_1$ moves a package from initial position $\mathcal{I}_1$ to final position $\mathcal{F}_1$, then picks up another package at initial position $\mathcal{I}_2$ to drop it off at final position $\mathcal{F}_2$, and finally goes to its final resting position $\mathcal{R}_1$. The manipulator at initial resting position $\mathcal{O}_2$ moves a package from initial position $\mathcal{I}_4$ to final position $\mathcal{F}_4$, then picks up another package at initial position $\mathcal{I}_3$ to drop it off at final position $\mathcal{F}_3$ and then moves to its final resting position $\mathcal{R}_2$. The paths of the two manipulators on the graph are highlighted in Figure 6 in blue and green, respectively. Figure 7 shows the manipulators paths and packages initial and final positions in the reference frame. Note that, since the package at initial position $\mathcal{I}_1$ has to be rotated by $90°$ to final position $\mathcal{F}_1$, in general, the space required for ensuring the collision-free manipulation of this package is larger than the convex hull of the union of the space covered by the package before and after its manipulation. Here, we overestimate the collision-free space for performing the manipulation from $\mathcal{I}_1$ to $\mathcal{F}_1$ with the smallest rectangle that contains the area of the conveyor belt that is occupied by the package during its manipulation and whose edges are aligned with the axes of our reference frame. In Figure 7, collision-free areas associated to manipulations are represented as blue or green rectangles depending on whether such manipulations are carried out by the first or the second robot, respectively.

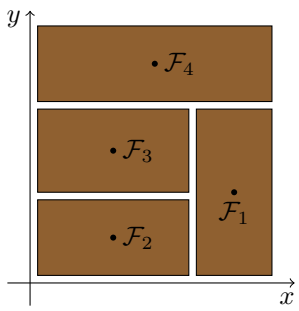


Fig. 5: Layout of three packages of different type.

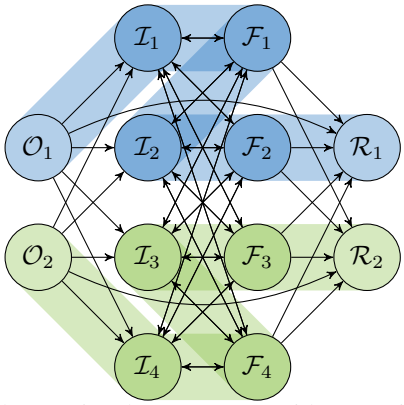


Fig. 6: Graph associated to a system with 2 manipulators and 3 packages with highlighted manipulators paths.

C. Packages positions and types

We define function $\gamma : \mathcal{I} \to \mathbb{R}^2$ such that, for $i \in \mathcal{I}$, $\gamma(i)$ represents the coordinates $(x_i, y_i) \in \mathbb{R}^2$ on the moving frame of the initial package position associated to node i. Function γ can

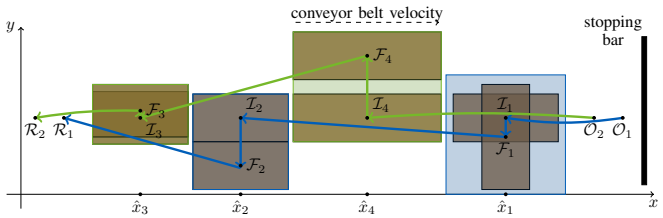


Fig. 7: Manipulations sequences on the conveyor belt.

also be extended to nodes in $\mathcal{F}$, however, in view of assumption 4) of Section II-A, until a final position is not associated to an initial one, its x-coordinate is not well-defined. Namely, if $j \in \mathcal{F}$ is a final package position that has been associated to initial position $i \in \mathcal{I}$, then $\gamma(j) = (x_i, y_j)$, where x_i is the x-coordinate of initial position i and y_j is the y-coordinate of final position j, according to the given layout. For any initial position $i \in \mathcal{I}$, quantity x_i (its coordinate along the x-axis) can be considered an optimization variable, as our goal is to obtain a positions sequence which is as short as possible, whilst the coordinate along the y-axis is known a priori as it corresponds to the y-coordinate of one of the available lines of the infeed conveyor. Since packages can have different shapes, we denote by $\mathcal{L}$ the set of different available package types and by $\mathcal{L}_{\mathcal{F}} : \mathcal{F} \to \mathcal{L}$ the function that associates to each final position $j \in \mathcal{F}$, the type of package $\mathcal{L}_{\mathcal{F}}(j)$ that occupies such position. Note that function $\mathcal{L}_{\mathcal{F}}$ is known a priori since the pallet layout is assigned.

D. Manipulators time-windows

Let q_m be the initial x-coordinate of the center of the m-th manipulator base, and let $-w$ be the speed at which the base moves along the x-axis (negative since the movement is in a direction opposite to the conveyor belt velocity). Then, at time t, the x-coordinate of the base is $q_m - tw$. We assume that the operating space of each manipulator is a rectangle aligned with the belt. On the x-axis, the rectangle is centered at the base of the manipulator and has length $2W$, with $W > 0$. Thus, with respect to the moving frame, at time t, the operating space of the m-th manipulator along the x-axis is the interval $[q_m - tw - W,\ q_m - tw + W]$. On the y-axis, it covers the whole width of the belt (see Figure 1). As manipulators bases move with speed $-w$ with respect to the reference frame, package positions can be visited by manipulators only within specific time windows. At time t, the m-th manipulator can visit node $k \in \mathcal{V}$ only if $\gamma(k)$ belongs to the m-th manipulator operating space. In other words, letting x be the x-coordinate of $\gamma(k)$, it must hold that $x \in [q_m - tw - W,\ q_m - tw + W]$ or, equivalently, $tw \in [q_m - x - W,\ q_m - x + W]$. Setting $[a_k^m, b_k^m] = [w^{-1}(q_m - x - W),\ w^{-1}(q_m - x + W)]$, for all $k \in \mathcal{V}$ and $m \in \mathcal{M}$, if the m-th manipulator visits node k at time τ_m, then it must hold that $\tau_m \in [a_k^m, b_k^m]$. This corresponds to a time window requirement on nodes visits.

E. Collisions handling

As mentioned earlier, manipulators do not lift packages from the belt. In order to ensure collisions avoidance, similarly to what we did in [5], we introduce the following quantities which depend on positions x-coordinates. For $i \in \mathcal{I}$, $j \in \mathcal{F}$, $k \in \mathcal{N}$, and $\ell \in \mathcal{L}$, define binary variables $c_{i,j}^{k,\ell}(x) \in \{0,1\}$ such that

$$c_{i,j}^{k,\ell}(x) = \begin{cases} 0, & \text{if a manipulation from } \gamma(i) \text{ to } \gamma(j) \\ & \text{collides with a package of type } \ell \\ & \text{placed at } \gamma(k), \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

In other words, $c_{i,j}^{k,\ell}(x) = 1$ if a package, moved from the initial position associated to node i to the position in the final layout associated to node j, does not collide with a package of type ℓ, placed at the position corresponding to node k. Otherwise, $c_{i,j}^{k,\ell}(x) = 0$.

We also need to introduce other quantities for ensuring the collision-free placement of a package on the conveyor belt from the infeed conveyor. For $i \in \mathcal{I}$, $k \in \mathcal{N}$, and $\ell, \bar{\ell} \in \mathcal{L}$, define binary variables $\bar{c}_{i,\ell}^{k,\bar{\ell}}(x) \in \{0,1\}$ such that

$$\bar{c}_{i,\ell}^{k,\bar{\ell}}(x) = \begin{cases} 0, \text{ if a package of type } \ell \text{ placed at } \gamma(i) \text{ col-} \\ \quad \text{-lides with one of type } \bar{\ell} \text{ placed at } \gamma(k), \\ 1, \text{ otherwise.} \end{cases} \tag{3}$$

Observe that, for $k \in \mathcal{F}$, values $c_{i,j}^{k,\ell}(x)$ and $\bar{c}_{i,\ell}^{k,\bar{\ell}}(x)$ are well defined only if $\ell = \mathcal{L}_{\mathcal{F}}(k)$. Note also that in the computation of $c_{i,j}^{k,\ell}(x)$, the type of package that is manipulated from initial position i to final position j is known, since it must correspond to $\mathcal{L}_{\mathcal{F}}(j)$.

F. Precedence relations on final positions

As already mentioned, the stopping bar placed at the end of the conveyor belt allows obtaining the final configuration by aligning packages along the x-axis (see Figure 1). To obtain the desired layout, we need to make sure that packages of different type, that occupy overlapping intervals on the y-axis in the final layout, are inserted on the belt at x-coordinates that respect the same ordering as in the final layout. We introduce a partial order relation over $\mathcal{F}$, denoted by $<_P$ such that for $j, k \in \mathcal{F}$, $j <_P k$ if the x-coordinate associated to the final position j is smaller than that associated to final position k. Recall that the conveyor belt moves from left to right, so that a position with a bigger x-coordinate precedes a position with a smaller x-coordinate. More precisely, we associate to each final position $j \in \mathcal{F}$ an interval $\mathcal{Y}(j)$ that contains the y-coordinates of the package that occupies final position j. If $j <_P k$, we say that final position k precedes final position j. Then $j <_P k$ if and only if:

- intervals associated to final positions j and k are overlapping, that is, $\mathcal{Y}(j) \cap \mathcal{Y}(k) \neq \varnothing$
- $x_j < x_k$.

As an example, consider the pallet layout of Figure 5 made out of four packages of different type. The package at final position $\mathcal{F}_1$ needs to precede packages at positions $\mathcal{F}_2$ and $\mathcal{F}_3$, that is, $\mathcal{F}_2 <_P \mathcal{F}_1$ and $\mathcal{F}_3 <_P \mathcal{F}_1$, since $\mathcal{Y}(\mathcal{F}_1) \cap \mathcal{Y}(\mathcal{F}_2) \neq \varnothing$, $\mathcal{Y}(\mathcal{F}_1) \cap \mathcal{Y}(\mathcal{F}_3) \neq \varnothing$, and $x_{\mathcal{F}_2} < x_{\mathcal{F}_1}$, $x_{\mathcal{F}_3} < x_{\mathcal{F}_1}$. Whilst the three packages at positions $\mathcal{F}_2$, $\mathcal{F}_3$ and $\mathcal{F}_4$ can be placed in either order with respect to each other, that is, $\mathcal{F}_i \not<_P \mathcal{F}_j$ and $\mathcal{F}_j \not<_P \mathcal{F}_i$, since $\mathcal{Y}(\mathcal{F}_i) \cap \mathcal{Y}(\mathcal{F}_j) = \varnothing$, for $i, j \in \{2,3,4\}$ and $i \neq j$ (placing one before the other will not compromise the correct formation of the layer). The same reasoning applies

to $\mathcal{F}_1$ and $\mathcal{F}_4$ for which $\mathcal{F}_1 \not\leftarrow_P \mathcal{F}_4$ and $\mathcal{F}_4 \not\leftarrow_P \mathcal{F}_1$, since $\mathcal{Y}(\mathcal{F}_1) \cap \mathcal{Y}(\mathcal{F}_4) = \varnothing$.

G. Decision variables and feasible set

The decision variables of our problem are:

- the x-coordinates of the available initial positions on the conveyor belt $\{x_i\}_{i\in\mathcal{I}} \in \mathbb{R}^I$,
- the integer variables that represent the assignment of packages to available initial positions on the conveyor belt: for each $i \in \mathcal{I}$, define $p_i \in \mathcal{L} \cup \{0\}$ such that

$$p_i = \begin{cases} \ell, & \text{if initial position } i \text{ is occupied} \\ & \text{by a package of type } \ell, \\ 0, & \text{otherwise.} \end{cases}$$

- the binary flow variables that represent the sequence of manipulations for each robot. Namely, for each $m \in \mathcal{M}$, for each $i, j \in \mathcal{V}$, binary variable $X^m_{i,j} \in \{0, 1\}$ is such that

$$X^m_{i,j} = \begin{cases} 1, & \text{if manipulator } m \text{ moves} \\ & \text{from node } i \text{ to node } j, \\ 0, & \text{otherwise.} \end{cases}$$

To simplify the notation, we will denote $\{x_i\}_{i\in\mathcal{I}}$, $\{p_i\}_{i\in\mathcal{I}}$, and $\{X^m_{i,j}\}_{i,j\in\mathcal{V},m\in\mathcal{M}}$ simply by x, p, and X, respectively. Let $\mathcal{B} \subseteq \mathbb{R}^I \times (\mathcal{L} \cup \{0\})^I \times \{0, 1\}^{|\mathcal{V}|\times|\mathcal{V}|\times M}$ be the feasible set of the 3PSP. Let

$$\begin{aligned} \mathcal{B}_x = \{&(p, X) \in (\mathcal{L} \cup \{0\})^I \times \{0, 1\}^{|\mathcal{V}|\times|\mathcal{V}|\times M} \mid \\ &(\exists x \in \mathbb{R}^I)\ (x, p, X) \in \mathcal{B}\}, \\ \mathcal{B}_{p,X} = \{&x \in \mathbb{R}^I \mid (\exists p, X \in (\mathcal{L} \cup \{0\})^I \times \\ &\{0, 1\}^{|\mathcal{V}|\times|\mathcal{V}|\times M})\ (x, p, X) \in \mathcal{B}\} \end{aligned} \tag{4}$$

be the discrete and continuous sections of $\mathcal{B}$, respectively. In other words, $\mathcal{B}_x$ is the set of feasible assignments p and flow variables X, for a given choice of positions x. Set $\mathcal{B}_{p,X}$ contains all feasible positions x, for a given choice of assignments p and flow variables X. The objective function is $F : \mathcal{B} \to \mathbb{R}$, $F(x, p, X) = \min_{i\in\mathcal{I}|p_i\neq 0}\{x_i\}$ that associates to a triplet $(x, p, X) \in \mathcal{B}$ the minimum x-coordinate. Note that $-F(x, p, X)$ is the length of the packages sequence (since all the x-coordinates are nonpositive). Our goal is to minimize the sequence length or, equivalently, maximize function F, that is,

$$\max_{(x,p,X)\in\mathcal{B}} F(x, p, X), \tag{5}$$

III. Solution strategy

In [5], we modeled the 3PSP as a MILP, but we noted that the direct solution of the complete problem requires a time that grows very quickly with the total number of packages. Indeed, even simple configurations with 5–6 packages require a computational time of a few hours. In most application of high-speed palletizing machines, manipulation sequences are precomputed before the machine installation, thus it is not necessary to solve the 3PSP in real-time. However, typical pallet layers can have more than 20–30 packages, and the solution of the complete MILP is not a viable option. Thus, in [5], we separated the 3PSP into two subproblems:

- In the first one, we assume that the initial position variables x are known, and we optimize packages assignment variables p and manipulator assignment variables X.
- In the second one, we assume that p and X are known, and we optimize initial position variables x.

Roughly speaking, the first subproblem is related to the assignment of manipulation tasks to robots and the association between packages and initial positions. The second subproblem is related to the optimization of the x-coordinates of the initial package positions.

In [5], we proposed an algorithm which solves these two subproblems iteratively, until the length of the packages sequence cannot be reduced any further.

The solution strategy is outlined in Algorithm 1, where, at line 7, $\epsilon > 0$ is a tolerance on objective function F.

Algorithm 1 Solution strategy

1: $\bar{x} := x^0$
2: **repeat**
3: $\quad(\bar{p}, \bar{X}) := \arg\max_{(p,X)\in\mathcal{B}_{\bar{x}}} F(\bar{x}, p, X)$
4: $\quad\mu := F(\bar{x}, \bar{p}, \bar{X})$
5: $\quad\bar{x} := \arg\max_{x\in\mathcal{B}_{\bar{p},\bar{X}}} F(x, \bar{p}, \bar{X})$
6: $\quad\xi := F(\bar{x}, \bar{p}, \bar{X})$
7: **until** $\xi - \mu < \epsilon$
8: **return** $(\bar{x}, \bar{p}, \bar{X})$

In this algorithm, $x^0 \in \mathbb{R}^I$ is an initial value for the packages initial x-coordinates. We can simply choose $x_0 = (0, -d_0, -2d_0, \ldots, -d_0(N-1))$, where d_0 is a sufficiently large positive constant such that the initial positions are spaced enough to guarantee the existence of a feasible solution for the problem at line 3. Indeed, a sufficiently large value of d_0 ensures that manipulators have enough time to move all packages and that these motions cause no collisions. We iterate the solution of the two subproblems, in which the output of one is fed as an input parameter to the other, until the optimization at line 5 improves the value of the objective function F by less than ϵ.

In this work, Algorithm 1 is still the overall solution strategy, as in [5], but we use different methods for solving the two subproblems at lines 3 and 5. Indeed, in [5] we modeled both problems as MILPs, while, here, we solve the subproblem at line 3 by DP, and the one at line 5 by a simple bisection procedure. In addition to the solution strategy we presented in this section, the 3PSP could be solved by means of alternative methods, more computationally demanding, which are briefly discussed in Appendix D.

IV. Manipulation and package assignments subproblem

As said in the previous section, we separate the 3PSP into two subproblems, and we solve the one associated to manipulation and package assignments by means of DP, assuming that initial package position x-coordinate variables $\{x_i\}_{i\in\mathcal{I}}$ are known input parameters. Note that if the values of x are known, $c^{k,\ell}_{i,j}(x)$ and $\bar{c}^{k,\bar{\ell}}_{i,\ell}(x)$ can be precomputed, so that they are also considered known problem parameters. To formulate such subproblem as a DP one, we represent the set of all

possible packages and manipulators configurations. In the initial configuration, the belt is empty, and the manipulators are at their initial resting positions in $\mathcal{O}$. We want to reach a final configuration in which all packages are at the corresponding positions in the final layout and all manipulators are at their final resting positions in $\mathcal{R}$. In order to achieve this, we can perform two kinds of operations: the insertion of a package on the belt and the movement of a manipulator. We want to find the sequence of these operations that allows obtaining the shortest packages sequence.

A. State space

We define the state space as $\Sigma = \mathcal{V}^M \times \mathbb{R}^M \times (\{0\} \cup \mathcal{L})^{I+N} \times (\{0\} \cup \mathcal{I})^N$. Each state is a quadruple $(\theta, \tau, \phi, \psi)$, in which the variables have the following meaning:

- $\theta = (\theta_1, \ldots, \theta_M) \in \mathcal{V}^M$ is such that θ_m is the node currently occupied by manipulator m;
- $\tau = (\tau_1, \ldots, \tau_M) \in \mathbb{R}^M$ is such that τ_m is the time at which manipulator m has reached its current node θ_m;
- $\phi = (\phi_1, \ldots, \phi_{I+N}) \in (\{0\} \cup \mathcal{L})^{I+N}$ is a vector representing the occupancy status of each initial and final package position. That is, for any $k \in \mathcal{N}$, $\phi_k = 0$ if position k is empty, or $\phi_k = \ell$, with $\ell \in \mathcal{L}$, if position k is occupied by a package of type ℓ;
- $\psi = (\psi_1, \ldots, \psi_N) \in (\{0\} \cup \mathcal{I})^N$ is a vector representing the association between initial and final packages positions. That is, for any $j \in \mathcal{F}$, $\psi_j = 0$ if position j is empty, or $\psi_j = i$, with $i \in \mathcal{I}$, if final position j is occupied by the package that was inserted on the conveyor belt at initial position i. ψ provides an important piece of information on the state of the system since, given assumption 4) of Section II-A, knowing that a final position has been occupied by a package that was at a certain initial position gives us the x-coordinate of that final position. Without ψ, we would not be able to assign an x-coordinate to any of the occupied final positions.

The initial state σ_0 is such that all manipulators are at their initial positions at initial time $t = 0$ and the conveyor belt is empty. Namely:

$$\sigma_0 = ((\mathcal{O}_1, \ldots, \mathcal{O}_M), 0_{1,M}, 0_{1,I+N}, 0_{1,N}). \tag{6}$$

As an example, consider the layout of Figure 5 with four packages of different type such that $(\forall i \in \{1, 2, 3, 4\})\ \mathcal{L}(\mathcal{F}_i) = i$. Consider also the sequence of Figure 7, and assume the first manipulator is in $\mathcal{F}_1$ and the second manipulator is in $\mathcal{I}_4$; then its corresponding state is $(\theta, \tau, \phi, \psi)$ with $\theta = (\mathcal{F}_1, \mathcal{I}_4)$, $\tau = (\tau_1, \tau_2)$, $\phi = (0, 2, 3, 4, 1, 0, 0, 0)$, since the package at initial position $\mathcal{I}_1$ has been moved to a final position, whilst the packages of type 2, 3 and 4 are still at initial positions $\mathcal{I}_2$, $\mathcal{I}_3$ and $\mathcal{I}_4$, respectively, and $\psi = (\mathcal{I}_1, 0, 0, 0)$, as final position $\mathcal{F}_1$ has been occupied by the package inserted at $\mathcal{I}_1$ and the remaining final positions $\mathcal{F}_2$, $\mathcal{F}_3$ and $\mathcal{F}_4$ are still empty.

B. Objective function

The objective is to minimize the length of the packages sequence on the conveyor belt, that corresponds to the smallest x-coordinate among the positions occupied by the packages. Hence, the objective function to minimize is $f : \Sigma \to \mathbb{R}$ with

$$f(\phi, \psi) = - \min_{i \in \mathcal{I} \mid \phi_i \neq 0 \vee (\exists j \in \mathcal{F})\, \psi_j = i} x_i, \tag{7}$$

while $f(\phi, \psi) = \infty$, if set $S = \{i \in \mathcal{I} \mid \phi_i \neq 0 \vee (\exists j \in \mathcal{F})\ \psi_j = i\}$ is empty. Here, S represents the subset of the initial positions $\mathcal{I}$ that have been assigned to packages in the current configuration (ϕ, ψ). Namely, S represents the subset of $\mathcal{I}$ consisting of those initial positions $i \in \mathcal{I}$ that are currently occupied by a package ($\phi_i \neq 0$) or that were previously occupied by a package that has already been moved to a final position ($(\exists j \in \mathcal{F})\ \phi_j = i$). The length of the sequence is equal to the most negative x-coordinate among the used initial positions, changed of sign due to our choice of reference frame. Since packages are moved only along the y-axis, manipulations do not change packages x-coordinates and do not alter the length of the sequence.

C. Expansion of current state

As said, we define two types of moves: the insertion of a package on the belt and the motion of a manipulator, described by two transition functions $\rho_1 : \Sigma \times \mathcal{I} \times \mathcal{L} \to \Sigma$ and $\rho_2 : \Sigma \times \mathcal{M} \times \mathcal{V} \to \Sigma$, respectively. We define two corresponding admissibility functions $\eta_1 : \Sigma \times \mathcal{I} \times \mathcal{L} \to \{0, 1\}$, $\eta_2 : \Sigma \times \mathcal{M} \times \mathcal{V} \to \{0, 1\}$, that are equal to 1 on allowed transitions and 0 on forbidden ones.

Let $\sigma' = \rho_1(\sigma, i, \ell)$ be the state obtained from σ by inserting a package of type ℓ at initial position i. Then, if $\sigma = (\theta, \tau, (\phi_1, \ldots, \phi_{I+N}), \psi)$ and $\sigma' = (\theta', \tau', (\phi'_1, \ldots, \phi'_{I+N}), \psi')$:

1) Times and positions of manipulators and final packages configurations do not change: $\theta' = \theta$, $\tau' = \tau$, $\psi' = \psi$.
2) A package of type ℓ is added to initial position i, while the other initial positions do not change: $\phi'_i = \ell$, $(\forall k \in \mathcal{N})\ k \neq i \Rightarrow \phi'_k = \phi_k$.

Further, we set $\eta_1(\sigma, i, \ell) = 0$ if and only if at least one of the following conditions holds:

3) The package is added to a non-empty position, that is $\phi_i \neq 0$, or to an initial position that is already associated to a final one, that is $(\exists j \in \mathcal{F})\ \psi_j = i$.
4) The package is assigned to a node whose x-coordinate is greater than the x-coordinate of a package that is already on the belt. That is, $(\exists k \in \mathcal{I})\ \phi'_k \neq 0 \wedge x_i > x_k$.
5) The package collides with a package already positioned on the conveyor belt: $(\exists k \in \mathcal{N})\ \phi_k \neq 0 \wedge \bar{c}_{i,\ell}^{k,\phi_k} = 0$.
6) All the packages of type ℓ required for the final layout have already been placed on the conveyor belt, that is $|\{\phi_k \in \mathcal{N} \mid \phi_k = \ell\}| = N_\ell$.

Condition 8) ensures that the sequence of packages insertions respects the temporal order: since the conveyor belt moves from left to right and the infeed conveyor is placed at the beginning of the belt on the left, the x-coordinate of every inserted package is not greater than that of all previously inserted ones.

Let $\sigma' = \rho_2(\sigma, m, k)$ be the state obtained from σ by moving manipulator m to node k. Note that the manipulator moves a package only if it goes from an

initial node to a final one. Function ρ_2 is such that, if $\sigma = ((\theta_1, \ldots, \theta_m), (\tau_1, \ldots, \tau_M), (\phi_1, \ldots, \phi_{I+N}), (\psi_1, \ldots, \psi_N))$ and $\sigma' = ((\theta'_1, \ldots, \theta'_m), (\tau'_1, \ldots, \tau'_M), (\phi'_1, \ldots, \phi'_{I+N}), (\psi'_1, \ldots, \psi'_N))$:

7) Times and positions of manipulators different from m remain the same: $(\forall m' \in \mathcal{M})\ m' \neq m \Rightarrow (\theta'_{m'} = \theta_{m'} \ \wedge \ \tau'_{m'} = \tau_{m'})$.
8) The position and time of manipulator m are updated. Due to the time-window constraint, if the visit time of the destination node k is lower than the beginning of the associated time window a_k^m, then it is set to a_k^m: $\theta'_m = k$, $\tau'_m = \max\{\tau_m + t_{\theta_m k}, a_k^m\}$, where $t_{\theta_m k}$ represents the (precomputed) time a manipulator needs to move from position θ_m to position k. More formally, t_{ij} is the value of function $t : \mathbb{R}^{2\times 2} \to \mathbb{R}_+$ which associates to coordinates $\gamma(i), \gamma(j) \in \mathbb{R}^2$ of positions i and j, respectively, the travel time $t(\gamma(i), \gamma(j))$ from position i to j denoted by t_{ij}. Note that function t can represent travel times of any given manipulators dynamics.
9) If the manipulator moves a package (i.e., it moves from an initial position to a final one), variables ϕ and ψ are updated accordingly: $\theta_m \in \mathcal{I} \Rightarrow (\phi'_{\theta_m} = 0 \wedge \phi'_k = 1 \wedge \ \psi'_k = \theta_m)$, whilst $(\forall i \in \mathcal{N})\ i \neq \theta_m \Rightarrow \phi'_i = \phi_i$ and $(\forall j \in \mathcal{F})\ j \neq k \Rightarrow \psi'_j = \psi_j$.

Moreover, $\eta_2(\sigma, m, k) = 0$ if and only if at least one of the following conditions holds:

10) The motion violates the graph adjacency matrix A of (1): $A_{\theta_m, k} = 0$.
11) Manipulators move to unoccupied initial positions or to occupied final positions, that is, $k \in \mathcal{I}$ and $\phi_k = 0$ or $k \in \mathcal{F}$ and $\psi_k \neq 0$.
12) The time-window constraint is violated for the destination node: $\tau'_m > b_k^m$.
13) The package collides with other packages during its motion: $\theta_m \in \mathcal{I}$ and there exists $i \in \mathcal{N}$ such that $\phi_i = \ell \neq 0$, and $c^{i,\ell}_{\theta_m, k} = 0$.
14) Precedence constraints are violated: there exists $j \in \mathcal{F}$ such that $\psi_j \neq 0$, that is, position j is occupied by a package, and either $j <_P k$ and $x_j > x_k$ or $k <_P j$ and $x_k > x_j$.
15) If $k \in \mathcal{F}$, the package type does not correspond to the one assigned to the final node: $\phi_{\theta_m} \neq \mathcal{L}_{\mathcal{F}}(k)$.

Denoting by $\mathcal{P}(\Sigma)$ the power set of Σ, we define function $\rho : \Sigma \to \mathcal{P}(\Sigma)$ such that $\rho(\sigma)$, the expansion of state σ, consists of all allowed transitions, namely:

$$\rho(\sigma) = \{\sigma' \in \Sigma \mid (\exists i \in \mathcal{I})(\exists \ell \in \mathcal{L})\ \sigma' = \rho_1(\sigma, i, \ell) \wedge \eta_1(\sigma, \sigma') = 1\} \ \cup \ \{\sigma' \in \Sigma \mid (\exists m \in \mathcal{M})(\exists k \in \mathcal{V})\ \sigma' = \rho_2(\sigma, m, k) \ \wedge \ \eta_2(\sigma, \sigma') = 1\}.$$

The set of accepted states $\mathcal{A}$ is the subset of Σ such that all final positions are occupied, initial positions are empty, and the manipulators are at their final resting positions. Namely,

$$\mathcal{A} = \{((\mathcal{R}_1, \ldots, \mathcal{R}_M), \tau, \phi, \psi) \in \Sigma \mid (\forall j \in \mathcal{F})\ \phi_j = \mathcal{L}_{\mathcal{F}}(j) \wedge (\forall i \in \mathcal{I})\ \phi_i = 0\}.$$

For $\sigma \in \Sigma$, and $n \in \mathbb{N}$ we denote by $\rho^n(\sigma) = \rho(\rho(\cdots \rho(\sigma) \cdots))$ the function obtained by iterating ρ n-times, while the full expansion set $\rho^*(\sigma)$ is $\rho^*(\sigma) = \cup_{n\in\mathbb{N}} \rho^n(\sigma)$, and define the set of accepted states obtained from σ as $\rho^{\mathcal{A}}(\sigma) = \rho^*(\sigma) \cap \mathcal{A}$.

Define the cost function $V : \Sigma \to \{\mathbb{R}, \infty\}$ so that $V(\sigma)$ is the minimum cost among the accepted states obtained from the full expansion of σ, that is:

$$V(\sigma) = \min_{\sigma' \in \rho^{\mathcal{A}}(\sigma)} f(\sigma'), \tag{8}$$

with f defined as in (7), while $V(\sigma) = +\infty$, if $\rho^{\mathcal{A}}(\sigma) = \varnothing$.

Then, solving our problem is equivalent to computing $V(\sigma_0)$.

V. Monotonic Dynamic Programming problems

In this section we introduce the class of monotonic DP problems, that includes the problem presented in Section IV. We present some solution algorithms and reduction techniques. In Section VI, we will apply these algorithms to the problem at hand.

Consider a generic DP problem: let Σ be the set of states, $\mathcal{A} \subset \Sigma$ the accepted states, $\rho : \Sigma \to \mathcal{P}(\Sigma)$ the expansion function, and $f : \Sigma \to \mathbb{R}$ the objective function. Given an initial state $\sigma_0 \in \Sigma$, we want to compute $\min_{\sigma \in \rho^{\mathcal{A}}(\sigma_0)} f(\sigma)$. We consider *monotonic* problems, that is, we assume that:

$$(\forall \sigma \in \Sigma) \min_{\sigma' \in \rho(\sigma)} f(\sigma') \geq f(\sigma),$$

in other words, the value of the objective function on states obtained by expanding σ is not lower than $f(\sigma)$. Note that function f in (7) satisfies this assumption.

A. Basic algorithm

First, we formulate a basic DP algorithm (Algorithm 2), which explores all states obtained by expanding the initial one. We define a priority queue Q that contains the states to be expanded. Namely, Q is an ordered set of pairs $(\sigma, f(\sigma)) \in \Sigma \times \mathbb{R}$, in which $\sigma \in \Sigma$ is a state and $f(\sigma)$ is the value of the objective function at σ. We perform two operations on Q: Enqueue$(Q, (\sigma, f(\sigma)))$, which inserts pair $(\sigma, f(\sigma))$ in Q, and Top(Q), which returns the state σ in Q with the highest priority, that is, with minimum value $f(\sigma)$, and removes pair $(\sigma, f(\sigma))$ from the queue.

Algorithm 2 Dynamic Programming

```
1: Initialization: U := +∞, Enqueue(Q, (σ_0, f(σ_0))).
2: State extraction: σ := Top(Q)
3: repeat
4:     State expansion: Σ' := ρ(σ)
5:     for all σ' ∈ Σ' | f(σ') < U do
6:         Enqueue(Q, (σ', f(σ')))
7:         if σ' ∈ A then
8:             U := f(σ')
9: until Q = ∅
```

At line 1, we add the initial state σ_0 to Q. The elements of the queue are in ascending order, according to the value of objective function f. We also set the initial value of the upper bound U to $+\infty$. At line 2, we extract the open state σ with the lowest value of f. At line 4, we compute the corresponding expansion set $\Sigma' = \rho(\sigma)$. At line 6, we add to the queue every element σ' of Σ such that $f(\sigma')$ is lower than the current upper

bound U. Finally, if σ' is an accepted state, at line 8, U is updated. At the end of the algorithm, U is the problem solution $V(\sigma_0)$. Algorithm 2 can be very inefficient since it explores all states $\hat{\sigma} \in \rho^*(\sigma)$ such that $f(\sigma) < V(\sigma_0)$. In the following, we introduce some cut (or reduction) strategies that reduce the set of explored states and the solution time.

B. Cuts based on lower bounds

A function $\hat{V} : \Sigma \to \mathbb{R}$ is a lower bound of V if $(\forall \sigma \in \Sigma)\ \hat{V}(\sigma) \leq V(\sigma)$. If an open state σ satisfies $\hat{V}(\sigma) \geq U$ (where U is the current upper bound) then σ can be discarded. This cut strategy is computationally convenient only if $\hat{V}$ is much faster to compute than V itself. Our goal is to efficiently compute a lower bound for each new state generated by a state expansion. We discard those states that cannot be expanded to a feasible solution or to one that improves the best solution found so far. In this way, we reduce the overall number of explored states.

In the following, we show that we can define a lower bound for V by defining a function $\Pi : \Sigma \to \hat{\Sigma}$, mapping the state space Σ to a set of reduced cardinality $\hat{\Sigma}$ and then, solving a problem on $\hat{\Sigma}$ by DP.

If the state space has the structure of a Cartesian product, that is, there exist sets Σ_1, Σ_2 such that $\Sigma = \Sigma_1 \times \Sigma_2$, then Π can be a simple projection from Σ to Σ_1 or Σ_2. In the general case, function Π can be arbitrary, however the cardinality of $\hat{\Sigma}$ should be much smaller than the cardinality of Σ, so that Π is not injective. In this way, each state $\hat{\sigma} \in \hat{\Sigma}$ could be the image of multiple states $\Pi^{-1}(\hat{\sigma}) \in \Sigma$.

We want to compute a lower bound on $V(\sigma)$, where $\sigma \in \Sigma$ is an assigned state. First, we introduce a restriction of the inverse of Π, depending on σ, as follows: $\Pi^+_\sigma : \hat{\Sigma} \to \mathcal{P}(\Sigma)$, with

$$\Pi^+_\sigma(\hat{\sigma}) = \Pi^{-1}(\hat{\sigma}) \cap \rho^*(\sigma).$$

In other words, $\Pi^+_\sigma(\hat{\sigma})$ is set of the counterimages with respect to Π of $\hat{\sigma}$, that are also expansions of state σ. Consider function $\Pi^+_\sigma\Pi : \Sigma \to \Sigma$. For $\hat{\sigma} \in \Sigma$, set $\Pi^+_\sigma\Pi(\hat{\sigma})$ represents all states belonging to the expansion of σ, that are projected by Π to the same reduced state $\Pi(\hat{\sigma})$.

We extend function $\Pi^+_\sigma\Pi : \Sigma \to \Sigma$ to a function $\Pi^+_\sigma\Pi : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$, by setting, for $A \subset \Sigma$, $\Pi^+_\sigma\Pi(A) = \bigcup_{\sigma' \in A} \Pi^+_\sigma\Pi(\sigma')$.

Function $\Pi^+_\sigma\Pi$ satisfies the following two properties.

Proposition V.1. *For any $\sigma \in \Sigma$:*

a) $(\forall A \subseteq \rho^*(\sigma))\ A \subseteq \Pi^+_\sigma\Pi(A)$,
b) $(\forall A, B \subseteq \Sigma)\ \Pi^+_\sigma\Pi(A \cap B) = \Pi^+_\sigma\Pi(A) \cap \Pi^+_\sigma\Pi(B)$.

Proof. a) $\Pi^+_\sigma\Pi(A) = \Pi_\sigma^{-1}\Pi(A) \cap \rho^*(\sigma) \supseteq A \cap \rho^*(\sigma) = A$, since $\Pi_\sigma^{-1}\Pi(A) \supseteq A$, by the definition of the inverse and $A \subseteq \rho^*(\sigma)$ by assumption.

b) $\Pi^+_\sigma\Pi(A \cap B) = \Pi_\sigma^{-1}\Pi(A \cap B) \cap \rho^*(\sigma) = \big(\Pi_\sigma^{-1}\Pi(A) \cap \rho^*(\sigma)\big) \cap \big(\Pi_\sigma^{-1}\Pi(B) \cap \rho^*(\sigma)\big) = \Pi^+_\sigma\Pi(A) \cap \Pi^+_\sigma\Pi(B)$.

□

In particular, property a) states that map $\Pi^+_\sigma\Pi : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$ is nondecreasing.

We define an expansion function for the reduced state $\hat{\rho}_\sigma : \hat{\Sigma} \to \mathcal{P}(\hat{\Sigma})$ such that

$$(\forall \sigma' \in \rho^*(\sigma))\ \hat{\rho}_\sigma(\Pi(\sigma')) \supseteq \Pi(\rho(\sigma')). \tag{9}$$

In other words, the expansion function for the reduced state must be sufficiently large. To satisfy (9), it is possible to set $\hat{\rho}_\sigma(\hat{\sigma}) = \Pi(\rho(\Pi^+_\sigma(\hat{\sigma})))$, or choose any $\hat{\rho}_\sigma$ such that $\hat{\rho}_\sigma(\hat{\sigma}) \supseteq \Pi(\rho(\Pi^+_\sigma(\hat{\sigma})))$.

Moreover, we choose an objective function for the reduced state $\hat{f}_\sigma : \hat{\Sigma} \to \mathbb{R}$ such that

$$(\forall \sigma' \in \rho^*(\sigma))\ \hat{f}_\sigma(\Pi(\sigma')) \leq f(\sigma'). \tag{10}$$

Namely, $\hat{f}_\sigma(\Pi(\sigma'))$ is a lower bound for $f(\sigma')$ on all elements σ' that are expansions of the initial state σ. For instance, it is possible to set $\hat{f}_\sigma(\hat{\sigma}) = \min_{\sigma' \in \Pi^+_\sigma(\hat{\sigma})} f(\sigma')$, or choose any function $\hat{f}_\sigma$ such that $\hat{f}_\sigma(\hat{\sigma}) \leq \min_{\sigma' \in \Pi^+_\sigma(\hat{\sigma})} f(\sigma')$.

Finally, setting $\hat{\mathcal{A}} = \Pi(\mathcal{A})$, we define the lower bound function $\hat{V} : \Sigma \to \mathbb{R}$ as $\hat{V}(\sigma) = \min_{\hat{\sigma} \in \hat{\rho}^*_\sigma(\Pi(\sigma)) \cap \hat{\mathcal{A}}} \hat{f}_\sigma(\hat{\sigma})$. Note that $\hat{V}$ can be computed by applying the DP algorithm presented in Algorithm 2 over $\hat{\Sigma}$.

The following proposition shows that $\hat{V}$ is a lower bound for V.

Proposition V.2. *For all $\sigma \in \Sigma$, $\hat{V}(\sigma) \leq V(\sigma)$.*

Proof.

$$\begin{aligned}
\hat{V}(\sigma) &= \min_{\hat{\sigma} \in \hat{\rho}^*_\sigma(\Pi(\sigma)) \cap \hat{\mathcal{A}}} \hat{f}_\sigma(\hat{\sigma}) \\
&\leq \min_{\hat{\sigma} \in \hat{\rho}^*_\sigma(\Pi(\sigma)) \cap \hat{\mathcal{A}}} \ \min_{\sigma' \in \Pi^+_\sigma(\hat{\sigma})} f(\sigma') \\
&= \min_{\sigma' \in \Pi^+_\sigma(\hat{\rho}^*_\sigma(\Pi(\sigma)) \cap \hat{\mathcal{A}})} f(\sigma') \\
&= \min_{\sigma' \in \Pi^+_\sigma(\hat{\rho}^*_\sigma(\Pi(\sigma)) \cap \Pi(\mathcal{A}))} f(\sigma') \\
&\leq \min_{\sigma' \in \Pi^+_\sigma(\Pi(\hat{\rho}^*_\sigma(\sigma)) \cap \Pi(\mathcal{A}))} f(\sigma') \\
&= \min_{\sigma' \in \Pi^+_\sigma(\Pi(\hat{\rho}^*_\sigma(\sigma) \cap \mathcal{A}))} f(\sigma') \\
&\leq \min_{\sigma' \in \hat{\rho}^*_\sigma(\sigma) \cap \mathcal{A}} f(\sigma') \\
&= \min_{\sigma' \in \rho^{\mathcal{A}}(\sigma)} f(\sigma') = V(\sigma),
\end{aligned}$$

where the first inequality is a consequence of Assumption (10) on $\hat{f}$, the second equality is just a rewriting of the previous statement, the third one follows from the definition of $\hat{\mathcal{A}}$, the second inequality is a consequence of Assumption (9) on $\hat{\rho}^*_\sigma$, the fourth equality follows from b) of Proposition V.1, the third inequality is a consequence of a) of Proposition V.1, and the last equality is given by (8). □

VI. Application to our problem

We consider the following two reductions which play an important role in improving the performances of Algorithm 2.

A. Omit packages configurations

In the following, for a vector $x \in \mathbb{R}^n$, we denote by $\|x\|_0$ the number of non-zero components of x. In the reduced state $\hat{\Sigma}$ we do not store the full packages occupation state, represented

by variables ϕ, ψ, but only the number of packages assigned to initial and final positions. Namely,

$$\Pi((\theta, \tau, \phi, \psi)) = (\theta, \tau, n_\phi, n_\psi),$$

where $n_\phi = \|\phi\|_0 - \|\psi\|_0$ is the number of packages that have been inserted on the belt, but not yet placed at their final position, and $n_\psi = \|\psi\|_0$ is the number of packages placed at their final position. Let $\sigma = (\theta, \tau, \phi, \psi)$ be the state on which we want to evaluate the lower bound and let $\hat{\sigma} = (\hat{\theta}, \hat{\tau}, n_\phi, n_\psi) \in \hat{\Sigma}$ be a generic reduced state.

We define $\hat{\rho}_\sigma : \hat{\Sigma} \to \mathcal{P}(\hat{\Sigma})$ as follows: $(\forall \hat{\sigma} \in \hat{\Sigma})$

$$\hat{\rho}_\sigma(\hat{\sigma}) = \{\hat{\sigma}' \in \hat{\Sigma} \mid (\exists m \in \mathcal{M})\ (\exists k \in \mathcal{V})\ \hat{\sigma}' = \hat{\rho}_{\sigma 2}(\hat{\sigma}, m, k) \wedge\ \hat{\eta}_{\sigma 2}(\hat{\sigma}, m, k) = 1\},$$

where transition function $\hat{\rho}_{\sigma 2} : \hat{\Sigma} \times \mathcal{M} \times \mathcal{V} \to \hat{\Sigma}$ plays the same role as transition function ρ_2 on the reduced states and is defined as follows. If $\hat{\sigma} = (\hat{\theta}, \hat{\tau}, n_\phi, n_\psi)$ and $\hat{\sigma}' = \hat{\rho}_{\sigma 2}(\hat{\sigma}, m, k) = (\theta', \tau', n'_\phi, n'_\psi)$, then $\hat{\sigma}'$ satisfies the same conditions 7) and 8), whilst condition 9) becomes

$$\theta_m \in \mathcal{I} \Rightarrow (n'_\phi = n_\phi - 1\ \wedge\ n'_\psi = n_\psi + 1).$$

Moreover, $\hat{\eta}_{\sigma 2}(\hat{\sigma}, m, k) = 0$ if and only if at least one of the following conditions, adapted from previous conditions 10)–15), is not satisfied, namely:

- $\hat{10}$) The motion violates the graph adjacency matrix: $A_{\theta_m, k} = 0$.
- $\hat{11}$) Manipulators move to unoccupied initial positions or to occupied final positions, according to the initial state σ, that is, if $k \in \mathcal{I}$ and $\phi_k = 0$ or if $k \in \mathcal{F}$ and $\psi_k \neq 0$.
- $\hat{12}$) The time-window constraint is violated for the destination node: $\tau'_m > b^m_k$.
- $\hat{13}$) The moved package collides with packages that already occupy their final positions in σ. That is, $\theta_m \in \mathcal{I}$ and there exists $j \in \mathcal{F}$ such that $\psi_j = \ell \neq 0$ and $c^{j,\ell}_{\theta_m, k} = 0$.
- $\hat{14}$) Precedence constraints are violated with respect to packages already assigned in $\mathcal{I}$ or $\mathcal{F}$, that is, there exists $j \in \mathcal{F}$ such that $\psi_j \neq 0$ at initial state σ and either $j <_P k$ and $x_j > x_k$ or $k <_P j$ and $x_k > x_j$.
- $\hat{15}$) If $k \in \mathcal{F}$, the package type does not correspond to the one assigned to the final node: $\phi_{\theta_m} \neq \mathcal{L}_\mathcal{F}(k)$ according to initial state σ.

We use this reduction only to test the feasibility of state σ. That is, if $\hat{\rho}^*_\sigma(\Pi(\sigma)) \notin \hat{\mathcal{A}}$ then $\rho^*(\sigma) \notin \mathcal{A}$, so that state σ can be removed. For this reason, objective function $\hat{f}_\sigma$ for the reduced problem is not relevant, and we can simply set it to the constant value $-\infty$ if $\hat{\rho}^*_\sigma(\Pi(\sigma)) \in \hat{\mathcal{A}}$, and $+\infty$ otherwise.

We call V_1 the corresponding cost function, that is, $\hat{V}_1(\sigma) = \min_{\hat{\sigma} \in \hat{\rho}^*_\sigma(\Pi(\sigma)) \cap \hat{\mathcal{A}}} \hat{f}_\sigma(\hat{\sigma})$.

B. Omit manipulator positions and times

In the reduced state we store only the packages configuration: $\Pi((\theta, \tau, \phi, \psi)) = (\phi, \psi)$.

The objective function for the reduced state is the same as (7), that is, $\tilde{f}(\phi, \psi) = f(\phi, \psi) = -\min_{i \in \mathcal{I} | \phi_i \neq 0 \vee (\exists j \in \mathcal{F})\ \psi_j = i} x_i$. Basically, $\tilde{\rho}_\sigma$ is defined in the same way as ρ, by considering only constraints 3), 4), 11), 13), 14) and 15). Namely, we define $\tilde{\rho}_\sigma : \hat{\Sigma} \to \mathcal{P}(\hat{\Sigma})$ as follows: $(\forall \hat{\sigma} \in \hat{\Sigma})\ \tilde{\rho}_\sigma(\hat{\sigma}) = \{\hat{\sigma}' \in \hat{\Sigma} \mid (\exists i \in \mathcal{I})\ (\exists \ell \in \mathcal{L})\ \hat{\sigma}' = \tilde{\rho}_{\sigma 1}(\hat{\sigma}, i, \ell) \wedge\ \tilde{\eta}_1(\hat{\sigma}, i, \ell) = 1\} \cup \{\hat{\sigma}' \in \hat{\Sigma} \mid (\exists i \in \mathcal{I})\ (\exists j \in \mathcal{F})\ \hat{\sigma}' = \tilde{\rho}_{\sigma 2}(\hat{\sigma}, i, j) \wedge\ \tilde{\eta}_2(\hat{\sigma}, i, j) = 1\}$. That is, in the reduced problem we substitute ρ_1, ρ_2 with $\tilde{\rho}_{\sigma_1}, \tilde{\rho}_{\sigma 2}$, and η_1, η_2 with $\tilde{\eta}_1, \tilde{\eta}_2$, respectively. Function $\tilde{\rho}_{\sigma 1} : \hat{\Sigma} \times \mathcal{I} \times \mathcal{L} \to \hat{\Sigma}$ is such that, if $\hat{\sigma} = (\phi, \psi)$ and $\tilde{\rho}_{\sigma 1}(\hat{\sigma}, i, \ell) = (\phi', \psi')$, condition 1) reduces to:

- $\tilde{1}$) $\psi' = \psi$,

and condition 2) does not change:

- $\tilde{2}$) $\phi'_i = \ell,\ (\forall k \in \mathcal{I})\ k \neq i \Rightarrow \phi'_k = \phi_k$.

Function $\tilde{\eta}_1 : \hat{\Sigma} \times \mathcal{I} \times \mathcal{L} \to \{0, 1\}$ is defined as before. Namely, $\tilde{\eta}_1(\hat{\sigma}, i, \ell) = 0$ if and only if at least one the following two conditions holds:

- $\tilde{3}$) $\phi_i \neq 0$.
- $\tilde{4}$) $(\exists k \in \mathcal{I})\ \phi'_k \neq 0 \wedge x_i > x_k$.

Since the reduced state does not include manipulators positions, we substitute the previous function $\tilde{\rho}_{\sigma 2}$ with a new transition function that considers only motions from initial to final positions. Function $\tilde{\rho}_{\sigma 2} : \hat{\Sigma} \times \mathcal{I} \times \mathcal{F} \to \hat{\Sigma}$ is such that, if $\tilde{\sigma} = (\phi, \psi)$ and $\tilde{\rho}_{\sigma 2}(\tilde{\sigma}, i, j) = (\phi', \psi')$: $\phi'_i = 0\ \wedge\ \phi'_j = \phi_i\ \wedge\ \psi'_j = i$.

Function $\tilde{\eta}_2 : \hat{\Sigma} \times \mathcal{I} \times \mathcal{F} \to \{0, 1\}$ is such that $\tilde{\eta}_2(\hat{\sigma}, i, j) = 0$ if and only if at least one of the following conditions holds:

- $\tilde{11}$) The package is moved from an unoccupied initial position or to an occupied final position, that is, if $\phi_i = 0$ or if $\phi_j \neq 0$.
- $\tilde{13}$) The package collides with other packages during its motion: $(\exists k \in \mathcal{I})\ \phi_k \in \mathcal{L}\ \wedge\ c^{k, \phi_k}_{i,j} = 0$ or $(\exists k \in \mathcal{F})\ \psi_k \neq 0\ \wedge\ c^{k, \mathcal{L}_\mathcal{F}(k)}_{i,j} = 0$.
- $\tilde{14}$) Precedence constraints are violated: there exists $k \in \mathcal{F}$ such that $k <_P j$ and $x_k > x_j$ or $j <_P k$ and $x_j > x_k$;
- $\tilde{15}$) The package type does not correspond to the one assigned to the final node: $\phi_i \neq \mathcal{L}_\mathcal{F}(j)$.

We call V_2 the cost function corresponding to this cut, that is, $\hat{V}_2(\sigma) = \min_{\hat{\sigma} \in \hat{\rho}^*_\sigma(\Pi(\sigma)) \cap \hat{\mathcal{A}}} \tilde{f}_\sigma(\hat{\sigma})$.

We can also eliminate some open states by using the following method.

C. Dominance

A state $\sigma = (\theta, \tau, \phi, \psi) \in \Sigma$ dominates $\sigma' = (\theta', \tau', \phi', \psi') \in \Sigma$ if both the following conditions hold:

- Manipulators are at the same nodes and the two states correspond to the same packages configurations, namely $\theta = \theta'$, $\phi = \phi'$, $\psi = \psi'$.
- The manipulator times of σ are not larger than those of σ': $\tau \leq \tau'$.

During state expansion, if we encounter a state σ' which is dominated by a state that is already in the priority queue, then σ' can be eliminated.

D. Enhanced algorithm

The previous discussion allows formulating Algorithm 3, the specialization of Algorithm 2 to the problem at hand. It uses lower bounds and dominance to reduce the set of expanded states.

Algorithm 3 Enhanced Dynamic Programming

1: Initialization: $U = +\infty$, Enqueue(Q, σ_0).
2: State extraction: $\sigma := \text{Top}(Q)$
3: **repeat**
4: State expansion: $\Sigma' := \rho(\sigma)$
5: **for all** $\sigma' = (\theta, \tau, \phi, \psi) \in \Sigma'$ **do**
6: **if** $\max \tau < U \wedge V_1(\sigma') < U \wedge V_2(\sigma') < U$
7: and the queue does not contain any configura-
8: tion $\tilde{\sigma} = (\theta, \tilde{\tau}, \phi, \psi)$ such that $\tilde{T} \leq T$ **then**
9: Enqueue(Q, σ')
10: **if** $\sigma' \in \mathcal{A}$ **then**
11: $U := \min\{U, f(\sigma')\}$
12: $\sigma^* := \sigma'$
13: **until** $Q = \varnothing$

VII. Position x-coordinates optimization subproblem

Let σ^* be the solution of DP obtained by Algorithm 3 and let $\sigma_0, \sigma_1, \ldots, \sigma_n = \sigma^*$ be the sequence of states leading to the optimal solution σ^*. That is, for each $h \in \{1, \ldots, n\}$, $\sigma_h \in \rho(\sigma_{h-1})$. From this sequence, we can find the corresponding values of manipulators task assignment variables X and packages assignment variables p, that will be fixed in the subproblem related to the optimization of initial positions x-coordinates. For any $\kappa \in \{0, \ldots, n\}$, set $\sigma_\kappa = (\theta^\kappa, \tau^\kappa, \phi^\kappa, \psi^\kappa)$. Then, we set $(\forall h \in \{1, \ldots, n\})$ $(\forall m \in \mathcal{M})$

$$X^m_{\theta^{h-1}_m, \theta^h_m} = 1 \iff \theta^h_m \neq \theta^{h-1}_m, \tag{11}$$

and $(\forall i \in \mathcal{I})$

$$p_i = \begin{cases} \phi^n_j, & \text{if } (\exists j \in \mathcal{F})\ \psi^n_j = i \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Let $\{s_\ell\}_{\ell \in \{1,\ldots,N\}}$ be such that, for $k \in \{1, \ldots, N-1\}$,

$$x_{s_{k+1}} \leq x_{s_k} \ \wedge\ (\exists j, j' \in \mathcal{F})\ \psi^n_j = s_k \wedge \psi^n_{j'} = s_{k+1}. \tag{13}$$

In other words, $\{s_\ell\}_{\ell \in \{1,\ldots,N\}}$ is the sequence of non-empty initial positions totally ordered in decreasing values of their x-coordinates. For each $k \in \{1, \ldots, N-1\}$, we call $d_k = x_{s_k} - x_{s_{k+1}}$ the distance between each pair of consecutive initial positions in sequence $\{s_\ell\}_{\ell \in \{1,\ldots,N\}}$, and optimize such distances $\{d_k\}_{k \in \{1,\ldots,N-1\}} \in \mathbb{R}^{N-1}_+$ one by one by bisection from $k = 1$ up to $k = N-1$. The bisection procedure is outlined in Algorithm 4, where ϵ is a tolerance (e.g., 1 mm), while $\alpha : \mathbb{R}^{N-1}_+ \to \mathbb{R}^I$ at lines 6 and 15 is such that $\alpha(d) = x$, with

$$\begin{cases} x_\iota = 0, & \iota \in \mathcal{I} \setminus \{s_\ell\}_{\ell \in \{2,\ldots,N\}} \\ x_{s_{k+1}} = x_{s_k} - d_k, & k \in \{1, \ldots, N-1\}. \end{cases} \tag{14}$$

Note that, for the purpose of performing a feasibility check, we only need to know the x-coordinates of initial positions that are occupied by a package, that is, we only need to know $\{\alpha(d)_{s_\ell}\}_{\ell \in \{1,\ldots,N\}}$. Since the x-coordinates of initial positions in $\mathcal{I} \setminus \{s_\ell\}_{\ell \in \{1,\ldots,N\}}$, (i.e., the empty ones) are irrelevant, we set them to 0.

Moreover, $\bar{f}$ at lines 7 and 12 is a feasibility check function that takes as input the current initial position x-coordinates (computed so far by the bisection procedure), the packages assignment to initial positions p given in (12), and the manipulations sequences X given in (11). It verifies whether the associated 3PSP is feasible or if some constraints are violated. Accordingly, it returns a boolean value b, true if the 3PSP is feasible and false otherwise.

Algorithm 4 Bisection

1: Input: p and X given as in (12) and (11), respectively.
2: **for** $k \in \{1, \ldots, N-1\}$ **do**
3: $\delta := \frac{x_{s_k} - x_{s_{k+1}}}{2}$
4: **while** $|\delta| > \epsilon$ **do**
5: $d_k := d_k - \delta$
6: $x := \alpha(d)$
7: $b := \bar{f}(x, p, X)$
8: **if** b is true **then**
9: $\delta := \frac{|\delta|}{2}$
10: **else**
11: $\delta := -\frac{|\delta|}{2}$
12: $b := \bar{f}(x, p, X)$
13: **if** b is false **then**
14: $d_k := d_k + |\delta|$
15: $x := \alpha(d)$
16: **return** $\{x_{s_k}\}_{k \in \{1,\ldots,N-1\}} \cup \{x_\iota\}_{\iota \in \mathcal{I} \setminus \{s_\ell\}_{\ell \in \{1,\ldots,N\}}}$

If b is true, then the procedure tries to further optimize current distance d_k by bisection (see line 9). Otherwise, if b is false, this means that distance d_k has been excessively reduced, and d_k is increased by bisection (see line 11).

Note that, when the condition at line 4 is false (i.e., when $|\delta| \leq \epsilon$), we need to perform an additional feasibility check (see line 12) since, if during the last execution of the while loop at line 4 boolean b is false, then we need to adjust distance d_k by increasing it in order to make it feasible again.

When optimizing distance d_k between two consecutive positions x_{s_k} and $x_{s_{k+1}}$, not only we are modifying the value of $x_{s_{k+1}}$, but also all coordinates $x_{s_{\bar{k}}}$ such that $\bar{k} \in \{k+1, \ldots, N-1\}$. Roughly speaking, we are pushing all coordinates smaller than x_{s_k} closer to it by the same distance.

Finally, note that at line 16, the set of x-coordinates returned by Algorithm 4 is the union between the x-coordinates of occupied positions $\{x_{s_\ell}\}_{\ell \in \{1,\ldots,N\}}$ and the set of x-coordinates $\{x_\iota\}_{\iota \in \mathcal{I} \setminus \{s_\ell\}_{\ell \in \{1,\ldots,N\}}}$ that remain unused. This second set of coordinates is determined by distributing in $[\min_k x_{s_k}, 0]$ the remaining $I - N$ x-coordinates in such a way that the minimum distance between any pair of x-coordinates along each infeed line is maximized.

A. Optimality result

Algorithm 4 does not guarantee attaining a globally optimal solution of the position x-coordinates optimization subproblem. Actually, it does not even guarantee finding a local minimum. However, the obtained solution satisfies a weaker optimality property, namely, it is a *lower basic solution*.

In this section, we make the following assumption.

Assumption VII.1. *We make an outer approximation of the space occupied during a manipulation with the minimum axis-aligned bounding box (AABB) associated to it.*

For further details we refer the reader to Appendix A. This assumption allows simplifying the computation of the collision variables $c_{i,j}^{k,\ell}$ and $\bar{c}_{i,\ell}^{k,\bar{\ell}}$. Also, the results presented in this section depend on this assumption, since it is used in the proof of Proposition VII.1 (see Appendix B).

Consider the following definitions, taken from [32].

Definition VII.1. *Let $N \subseteq \mathbb{R}_+^n$, then N is called normal if* $(\forall x \in N)\ (\forall x' \in \mathbb{R}_+^n)\ x' \leq x \Rightarrow x' \in N$.

Definition VII.2. *Let $R \subseteq \mathbb{R}_+^n$, then R is called reverse normal if* $(\forall x \in R)\ (\forall x' \in \mathbb{R}_+^n)\ x' \geq x \Rightarrow x' \in R$.

Consider the subset of $x \in \mathbb{R}^n$ such that

$$\begin{cases} g(x) \leq 1 \\ h(x) \geq 1, \end{cases} \tag{15}$$

where $g, h : \mathbb{R}_+^n \to \mathbb{R}$ are assigned increasing functions. If we set $N = \{x \in \mathbb{R}_+^n \mid g(x) \leq 1\}$ and $R = \{x \in \mathbb{R}_+^n \mid h(x) \geq 1\}$, we can rewrite system (15) as $x \in N \cap R$, where N is a normal set and R is a reverse normal one.

Definition VII.3. *Let $x \in N \cap R$, x is a lower basic solution of system* (15) *if* $(\forall x' \in N \cap R)\ x' \leq x \Rightarrow x' = x$.

Given sequence $\{s_k\}_{k\in\{1,\dots,N\}}$, defined as in (13), let $\{d_k\}_{k\in\{1,\dots,N-1\}}$, with $d_k = x_{s_k} - x_{s_{k+1}}$, be the sequence of distances between consecutive initial positions $\{x_{s_k}\}_{k\in\{1,\dots,N\}}$ along the x-axis, and let $g : \mathbb{R}_+^{N-1} \to \mathbb{R}$ be such that

$$g(d) = \frac{1}{|\min_{i\in\mathcal{I}} x_i^0|} \sum_{k=1}^{N-1} d_k, \tag{16}$$

where x^0 is the feasible initial value for x-coordinates given in Algorithm 1, and define $\mathcal{C} = \{d \in \mathbb{R}_+^{N-1} \mid g(d) \leq 1\}$. Note that function g is increasing, indeed, $(\forall d, d' \in \mathbb{R}_+^{N-1})\ d \leq d' \Rightarrow g(d) \leq g(d')$, and set $\mathcal{C}$ is normal.

Let (p, X) be the solution provided by the DP procedure at line 3 of Algorithm 1. Define

$$\mathcal{D} = \{d \in \mathbb{R}_+^{N-1} \mid (\alpha(d), p, X) \in \mathcal{B}\}, \tag{17}$$

and set $h : \mathbb{R}_+^{N-1} \to \mathbb{R}$ as

$$h(d) = \begin{cases} 1, & d \in \mathcal{D} \\ 0, & d \notin \mathcal{D} \end{cases} \tag{18}$$

Proposition VII.1. *For assigned values of p and X, set $\mathcal{D}$ defined in* (17) *is reverse normal.*

For a proof of Proposition VII.1 see Appendix B.

Let us now state the following proposition (for a proof see [32]).

Proposition VII.2. *An increasing function $f : \mathbb{R}_+^n \to \mathbb{R}$ achieves its minimum over $N \cap R$ at a lower basic solution of system* (15).

Definition VII.4. *Let $x \in N \cap R$, x is an ϵ-lower basic solution of system* (15) *if* $(\forall x' \in N \cap R)\ x' \leq x \Rightarrow \|x' - x\|_\infty \leq \epsilon$.

The following proposition shows that the solution provided by the bisection procedure cannot be improved by optimizing components of d one by one by quantities larger than ϵ.

Proposition VII.3. *The procedure outlined in Algorithm 4 provides an ϵ-lower basic solution for the 3PSP for fixed values of (p, X).*

For a proof of Proposition VII.3 see Appendix C.

VIII. Numerical experiments

We present some numerical results on randomly generated 3PSP instances. We implemented the DP algorithm and the bisection procedure in C++, and run the experiments on a 2.7 GHz Intel Core i5 dual-core with 8 GB of RAM. We compared the computational times of the DP approach against those obtained through the method presented in [5], extended for handling packages of different type and a multiple-line infeed conveyor. However, the problem solved in this work and in [5] do not coincide exactly. In fact, in [5], we approximated the travel time of manipulators movements as linear functions of the coordinates of initial and final positions. We did this in order to reduce the 3PSP to a MILP problem. Whereas, in the DP approach, manipulator motion times need not be linear functions and can be computed exactly. For this reason, the two approaches lead to slightly different solutions and, potentially, a feasible solution of one approach could be infeasible for the other one. However, to the best of our knowledge, the approach presented in [5] is the closest and only one we can compare the DP approach with. The model of [5] is solved with Gurobi [25], and we refer to it as MILP-G. Figure 9 presents some numerical results on solution times of the DP approach and MILP-G. We ran simulations for 19 different values of N (the number of packages), linearly spaced between 3 and 21. For each of them, we ran 20 simulations by randomly generating layouts for a 2.5×2 m pallet. We considered three types of packages (0.3×0.2 m, 0.6×0.2 m, and 0.3×0.4 m) and evenly distributed them among the available packages. For each test, we randomly placed non-overlapping packages either parallel or perpendicular to the conveyor belt velocity, in order to generate a random layout, as in Figure 8. For both the MILP-G and the DP approaches, we assumed to have $M = 2$ manipulators and considered a conveyor belt speed of $w = 0.45\ \mathrm{m\,s^{-1}}$. In the DP approach, we computed the transition times assuming that the manipulators followed a uniformly accelerated linear motion, followed by a uniform linear motion (in case manipulators manage to reach their maximum speed), and finally followed by a uniformly decelerated linear motion with a maximum acceleration/deceleration of $8\ \mathrm{m\,s^{-2}}$ and a maximum speed of $1.6\ \mathrm{m\,s^{-1}}$. Whilst, for the MILP-G, we assumed that the manipulators instantaneous acceleration was unbounded, that their speed was $\omega = 1.5\ \mathrm{m\,s^{-1}}$ and that transition times were computed as follows: for any $i, j \in \mathcal{V}$, let $\gamma(i) = (x_i, y_i)$ and $\gamma(j) = (x_j, y_j)$, then $t_{ij} = (|x_i - x_j| + |y_i - y_j|)/\omega$. In the two approaches, the use of different manipulators dynamic models naturally leads to different solutions. In general, the solution obtained with one dynamic model can be infeasible with respect to the other one, and viceversa. The dynamic model employed in the DP approach represents quite accurately the actual manipulators dynamics. On the other hand, in the MILP-G approach, we were forced to simplify the model to formulate the overall problem as a MILP. In general, the DP approach is more flexible, and allows adopting the dynamic model that

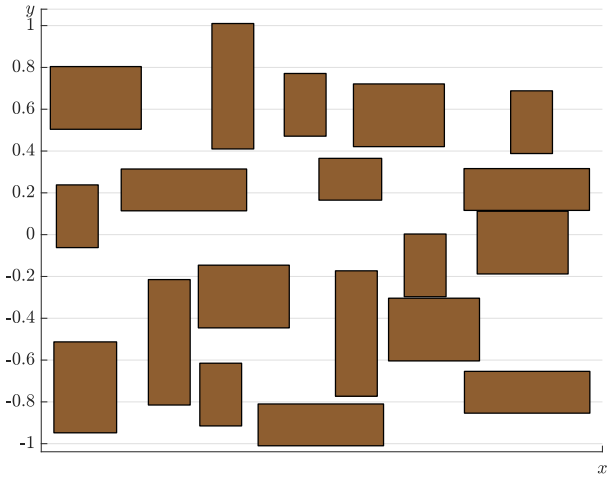


Fig. 8: Random layout with 19 packages of three different types.

better describes the manipulators with which the machine is equipped. Also, we assumed that the palletizing system had a 3-line infeed conveyor. Figure 9 shows the box-and-whisker plot of the solution times with the DP approach, together with its mean solution times, and those with MILP-G. Note that, after 15 packages, we stopped testing MILP-G, as it was clear that the DP approach was outperforming it. The DP approach was found to be on average 57.24 times faster than MILP-G for $N \in \{3, \dots, 15\}$, with a peak of 88.74 for tests with $N = 8$. Observe that the computational times shown in Figure 9 refer to the time required by each approach for converging to a feasible solution of the 3PSP. As already mentioned, these solutions do not coincide in general. Note that for both approaches the returned solution is guaranteed to be feasible but not even locally optimal. The two approaches are based on a variable-splitting approach. That is, we update the position coordinates, and then, we solve the manipulation assignment problem with *fixed* position coordinates. Since the latter subproblem is itself NP-hard (as it can be seen as a variant of a vehicle routing problem, see for instance [33]), detecting an optimal solution of the full problem quickly leads to a combinatorial explosion and to unacceptably high computational times, as already shown in [5]. However, for the DP approach we can at least prove that the obtained results are ϵ-lower basic solutions (see Proposition VII.3).

We also performed a set of numerical experiments to show the performance improvement due to the cut strategies introduced in Section VI. For different numbers of packages, ranging from 3 to 15, we solved 20 instances of the 3PSP on randomly generated layouts, for a total of 260 tests. We used the DP approach, both with and without the aforementioned cut strategies. Figure 10 displays the box-and-whisker plot of the computational times for the DP, with and without cuts. As the number of packages increases, so does the gap between the computational times of DP with and without cuts. On average, the performance improvement is 62.53%, with an increasing trend that peaks at 91.59% for layouts with 15 packages.

Finally, let us consider a system with two manipulators and a 2-line infeed conveyor, with the same previous manipulators parameters. Figure 11 shows the desired final layout. It is composed of 9 packages of three different types: type 1 (0.2×0.3 m), type 2 (0.4×0.3 m), and type 3 (0.8×0.3 m). Final positions are such that $(\forall j \in \{2, 8\})\ \mathcal{L}_{\mathcal{F}}(\mathcal{F}_j) = 1$, $(\forall j \in \{1, 3, 7, 9\})\ \mathcal{L}_{\mathcal{F}}(\mathcal{F}_j) = 2$ and $(\forall j \in \{4, 5, 6\})\ \mathcal{L}_{\mathcal{F}}(\mathcal{F}_j) = 3$, so that we have 2 packages of type 1, 4 packages of type 2, and 3 packages of type 3. We solved this problem with the DP approach. Figures 12 and 13 show the manipulation sequences for the first and the second manipulator, respectively. Black circles represent available initial package positions, which can be either occupied by a package or not. Labels $\mathcal{O}_1$ and $\mathcal{O}_2$ identify manipulators origin positions, whilst $\mathcal{R}_1$ and $\mathcal{R}_2$ correspond to manipulators final resting positions. The first manipulator follows the blue path, that starts from initial position $\mathcal{O}_1$ and ends at resting location $\mathcal{R}_1$, visiting the positions represented by blue circles. The blue boxes are the AABB outer approximations of the space occupied by packages during repositioning. Similarly, the second manipulator follows the green path, from $\mathcal{O}_2$ to $\mathcal{R}_2$, visiting the positions represented by green circles and with green AABBs for collision avoidance. Since we considered a 2-line infeed conveyor and a 9-package layout, the available initial package positions are 18, labeled with $\mathcal{I}_1, \dots, \mathcal{I}_{18}$, whilst final package positions are labeled with $\mathcal{F}_1, \dots, \mathcal{F}_9$. Note that it occurs that $\mathcal{R}_1 \equiv \mathcal{R}_2 \equiv \mathcal{F}_8$. Moreover, in Figure 12 the space occupied by the package moved from $\mathcal{I}_2$ to $\mathcal{F}_2$ overlaps with the package at $\mathcal{I}_4$. However, by the time the first robot performs such manipulation, the second robot has already moved the package at $\mathcal{I}_4$ to $\mathcal{F}_3$, hence the manipulation from $\mathcal{I}_2$ to $\mathcal{F}_2$ is collision free. The DP solution time for the previous example was 5.89 s, whilst the MILP-G solution time was 763.66 s.

Additional data and plots of other ∼13 000 random tests in total for the DP approach are available on OSF website at osf.io/yzexk. We generated 130 random layouts and considered 4 values of the conveyor belt speed, 5 values of the manipulators speed, and 5 values of the manipulators acceleration. For each randomly generated layout, and for each value of the conveyor belt speed, the manipulators speed, and the manipulators acceleration, we solved the associated problem through the DP approach.

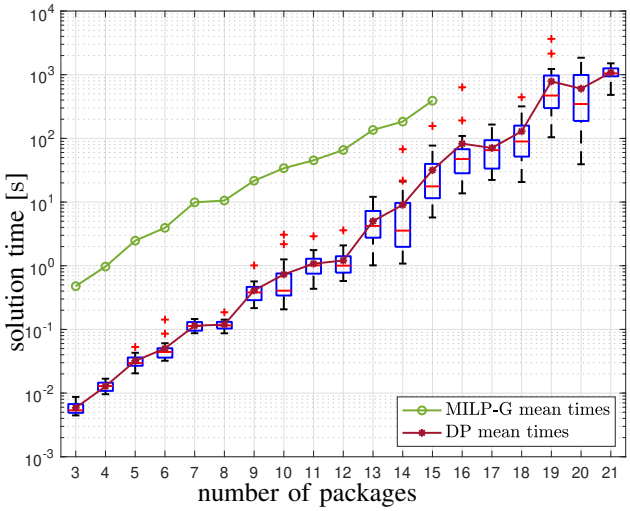


Fig. 9: Solution times of the DP approach compared to MILP-G for different numbers of packages.

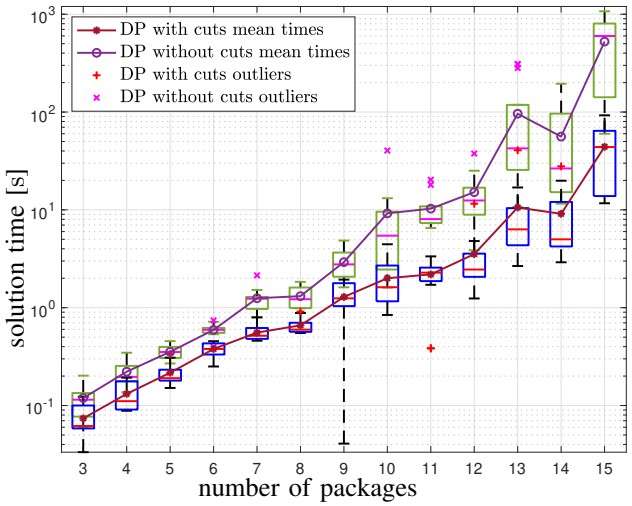


Fig. 10: Solution times of the DP approach with and without cuts for different numbers of packages.

IX. Conclusion and Future Research

We presented a DP-based approach for solving the 3PSP shortest packages sequence and the corresponding robot manipulations. By the numerical experiments on randomly generated problems, we showed the advantage of the DP approach over the MILP one in terms of computational times. The DP approach also allows for a higher freedom in modeling manipulators dynamics.

A possible development of this work would be allowing manipulators to modify packages x-coordinates with movements that are either parallel to the x-axis, diagonal (i.e., which modify at the same time both x and y-coordinates of packages), or similar to the knight's move in chess: manipulators could either first move packages along y-axis and then along x-axis, or viceversa. This last movement could be useful in case a diagonal movement would cause a collision. The use of such movements not only could improve the overall length of the sequence, hence, increasing the throughput of the palletizing machine, but it could also make the use of stopping bars unnecessary, allowing for a simpler machine design. Another development could involve the manipulation of packages in multiple steps: a package is first moved to an intermediate position by a manipulator and then moved to its final position by another one.

Both these developments would enormously increase the level of complexity of the problem. Such improvements would require not only a more involved collisions handling but a different state space definition and expansion. Computational times may be extremely high and the design of tailored cuts for reducing the state space exploration would be crucial for obtaining reasonable solution times.

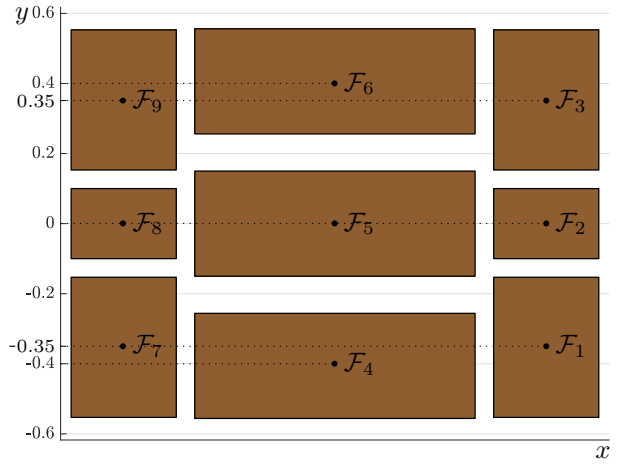


Fig. 11: Layout with 9 packages of three different types.

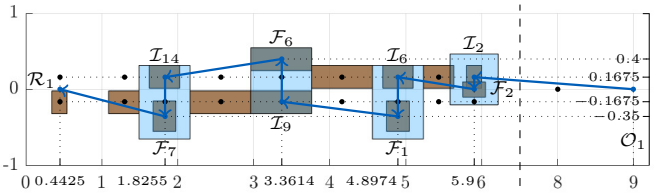


Fig. 12: Manipulations sequence of the first manipulator.

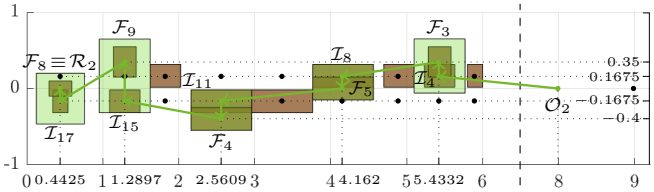


Fig. 13: Manipulations sequence of the second manipulator.

Appendix

A. AABBs for collision avoidance

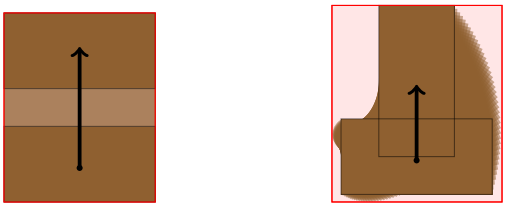
An AABB for a portion of space is a rectangle containing such space and such that its edges are aligned with the axes of the reference system. This is a convenient choice since collision detection between pairs of AABBs is particularly simple and fast. As an example, Figure 14a shows the AABB associated to a pure translation of a package, whilst Figure 14b shows the AABB associated to a rigid transformation of a package, that is, a translation combined with a rotation.

Note that, when performing a rigid transformation, we assume that the translation and the rotation are simultaneous (i.e., they start and end at the same time). Moreover, AABBs associated to packages exactly represent them since packages are themselves axis-aligned rectangles. Now, with respect to (2), consider an initial position i, a final position j, a position k, and a package of type ℓ. Let $\lambda_x, \lambda_y > 0$ be the semi-lengths of the edges parallel and perpendicular to the x-axis of the AABB associated to the manipulation of a package from position i to j, respectively. Moreover, let $\ell_x, \ell_y > 0$ be the semi-lengths of the edges parallel and perpendicular to the x-axis of the package of type ℓ at position k, respectively. Then, definition (2) can be rewritten as follows

$$c_{i,j}^{k,\ell}(x) = (x_i - \lambda_x > x_k + \ell_x) \vee (x_i + \lambda_x < x_k - \ell_x) \vee \\ \vee (y_i - \lambda_y > y_k + \ell_y) \vee (y_i + \lambda_y < y_k - \ell_y). \quad (19)$$

Similarly, with respect to (3), consider an initial position i, a position k, a package of type ℓ and one of type $\bar{\ell}$. Let $\ell_x, \ell_y > 0$ be the semi-lengths of the edges parallel and perpendicular to the x-axis of the package of type ℓ at position i, respectively. Moreover, let $\bar{\ell}_x, \bar{\ell}_y > 0$ be the semi-lengths of the edges parallel and perpendicular to the x-axis of the package of type $\bar{\ell}$ at position k, respectively. Then, definition (3) can be

rewritten as follows $\bar{c}^{k,\bar{\ell}}_{i,\ell}(x) = (x_i - \ell_x > x_k + \bar{\ell}_x) \vee (x_i + \ell_x < x_k - \bar{\ell}_x) \vee (y_i - \ell_y > y_k + \bar{\ell}_y) \vee (y_i + \ell_y < y_k - \bar{\ell}_y)$.



(a) Pure translation. (b) Rigid transformation.

Fig. 14: AABBs associated to different types of manipulations.

B. Proof of Proposition VII.1

Before proving Proposition VII.1, we need the following preliminary result on collision variables.

Proposition A.1. *Let (p, X) be a feasible pair of package and manipulation associations for the 3PSP, let function g be defined as in* (14)*, and let $d, d' \in \mathbb{R}^{N-1}$ be such that $d \leq d'$. Then, collision variables $c^{k,\ell}_{i,j}(\alpha(d)), \bar{c}^{k,\bar{\ell}}_{i,\ell}(\alpha(d))$ and $c^{k,\ell}_{i,j}(\alpha(d')), \bar{c}^{k,\bar{\ell}}_{i,\ell}(\alpha(d'))$ are such that, $(\forall i \in \mathcal{I})(\forall j \in \mathcal{F})(\forall k \in \mathcal{N})(\forall \ell \in \mathcal{L})$ $c^{k,\ell}_{i,j}(\alpha(d)) \leq c^{k,\ell}_{i,j}(\alpha(d'))$, and $(\forall i \in \mathcal{I})(\forall k \in \mathcal{N})(\forall \ell, \bar{\ell} \in \mathcal{L})$ $\bar{c}^{k,\bar{\ell}}_{i,\ell}(\alpha(d)) \leq \bar{c}^{k,\bar{\ell}}_{i,\ell}(\alpha(d'))$.*

Proof. Consider an initial position i, a final position j, a position k and a package of type ℓ, let $\lambda_x, \lambda_y, \ell_x, \ell_y$ be defined as in (19) and let $x = \alpha(d)$ and $x' = \alpha(d')$. Then, since $|x_i - x_k| \leq |x'_i - x'_k|$, by (19), we have that

$$\begin{aligned} c^{k,\ell}_{i,j}(x) = &(x_i - \lambda_x > x_k + \ell_x) \vee (x_i + \lambda_x < x_k - \ell_x) \vee \\ &\vee (y_i - \lambda_y > y_k + \ell_y) \vee (y_i + \lambda_y < y_k - \ell_y) \leq \\ \leq &(x'_i - \lambda_x > x'_k + \ell_x) \vee (x'_i + \lambda_x < x'_k - \ell_x) \vee \\ &\vee (y_i - \lambda_y > y_k + \ell_y) \vee (y_i + \lambda_y < y_k - \ell_y) = c^{k,\ell}_{i,j}(x'), \end{aligned}$$

that is, $c^{k,\ell}_{i,j}(x) \leq c^{k,\ell}_{i,j}(x')$. The same reasoning applies to $c^{k,\bar{\ell}}_{i,\ell}(x)$ and $c^{k,\bar{\ell}}_{i,\ell}(x')$. □

We can now prove Proposition VII.1.

Proof. Let $d \in \mathcal{D}$ and $d' \geq d$, we set new x-coordinate variables $x' = \alpha(d')$, with g defined as in (14). Obviously, conditions 3), 4), 6), 10), 11), 15) do not occur since they only depend on p and X, which are feasible by hypothesis. Moreover, for $k \in \{2, \dots, I\}$, set times $\tau'_k = \sum_{\bar{k}=1}^{k-1} \frac{d'_{\bar{k}-1} - d_{\bar{k}-1}}{w}$, and $\tau'_1 = 0$. Then, for all $m \in \mathcal{M}$, for all $k \in \mathcal{I}$, we set new time windows $a'^m_{s_k} = a^m_{s_k} + \tau'_k$, $b'^m_{s_k} = b^m_{s_k} + \tau'_k$, and visit times $t'_{s_k} = t_{s_k} + \tau'_k$, so that condition 12) never occurs, that is, $(\forall i, j \in \mathcal{V})$ $(\forall m \in \mathcal{M})$ $X^m_{i,j} = 1 \Rightarrow t'_i, t'_j \in [a'^m_i, b'^m_i]$. Condition 8) also holds true since $(\forall i, j \in \mathcal{V})$ $(\forall m \in \mathcal{M})$ $X^m_{i,j} = 1 \Rightarrow (\exists \tau_{k_i} \in \mathbb{R}_+)$ $t'_j = t_j + \tau_{k_i} \geq t_i + t_{ij} + \tau_{k_i} = t'_i + t_{ij}$. Now, considering condition 14), we have that if $j <_P k$ then, since $x_{i_j} < x_{i_k}$, there exist $s_{h_j}, s_{h_k} \in \{s_{\bar{k}}\}_{\bar{k} \in \{1,\dots,N-1\}}$ such that $x_{i_j} = x_{s_{h_j}}$ $x_{i_k} = x_{s_{h_k}}$ with $h_j > h_k$, hence, it holds that $x'_{i_j} = x_{i_j} + \sum_{\bar{k}=1}^{h_j-1}(d_{\bar{k}} - d'_{\bar{k}}) < x_{i_k} + \sum_{\bar{k}=1}^{h_j-1}(d_{\bar{k}} - d'_{\bar{k}}) < x_{i_k} + \sum_{\bar{k}=1}^{h_k-1}(d_{\bar{k}} - d'_{\bar{k}}) = x'_{i_k}$, that is, condition 14) never occurs. Finally, since $d' \geq d$, by Proposition A.1, new collision parameters are such that $c^{k,\ell}_{i,j}(\alpha(d')) \geq c^{k,\ell}_{i,j}(\alpha(d))$ and $\bar{c}^{k,\bar{\ell}}_{i,\ell}(\alpha(d')) \geq \bar{c}^{k,\bar{\ell}}_{i,\ell}(\alpha(d))$, that is, conditions 5) and 13) never occur. Hence, $x' \in \mathcal{B}_{p,X}$, which means that $d' \in \mathcal{D}$. □

C. Proof of Proposition VII.3

Proof. Let $d \in \mathcal{C} \cap \mathcal{D}$ be the solution provided by the bisection procedure and, by contradiction, let $\bar{d} \in \mathcal{C} \cap \mathcal{D}$ be such that $\bar{d} \leq d \ \wedge \ \|\bar{d} - d\|_\infty > \epsilon$. This would mean that $(\exists k' \in \{1, \dots, N-1\})$ $|\bar{d}_{k'} - d_{k'}| > \epsilon$. Now, since $h(\bar{d}) = 1$ and $\mathcal{D}$ is reverse normal, Algorithm 4 cannot return a value of $d_{k'}$ such that $d_{k'} > \bar{d}_{k'} + \epsilon$. So, $|\bar{d}_{k'} - d_{k'}| \leq \epsilon$, which contradicts the initial assumption. This means that $(\forall d' \in \mathcal{C} \cap \mathcal{D})$ $d' \leq d \Rightarrow \|d' - d\|_\infty \leq \epsilon$, that is, d is an ϵ-lower basic solution of (15). □

D. Alternative solution methods

In general, the solution strategy presented in Section III does not allow finding a global optimum of 3PSP, but provides good quality solutions with low computational time (see Section VIII). Here, we briefly present three alternative solution methods.

The first two methods exploit the concepts of monotonic optimization, recalled in Section VII-A. Define function $\tilde{\alpha} : \mathbb{R}^{I-1}_+ \to \mathbb{R}^I$, such that, if $x = \tilde{\alpha}(d)$, $x_1 = 0$ and, for $i \in \{1, \dots, I-1\}$, $x_{i+1} = x_i + d_i$.

Set

$$\tilde{\mathcal{D}} = \{d \in \mathbb{R}^{I-1}_+ \mid x = \tilde{\alpha}(d), \ \mathcal{B}_x \neq \varnothing\}, \tag{20}$$

where g is defined in (14) and $\mathcal{B}_x$ is defined in (4). In other words, $d \in \mathcal{D}$ if and only if $\mathcal{B}$ contains a feasible solution in which the initial positions are given by $\tilde{\alpha}(d)$.

The following proposition is analogous to Proposition VII.1 and can be proved in the same way.

Proposition A.2. *Set $\tilde{\mathcal{D}}$, defined in* (20)*, is reverse normal.*

Define $\tilde{F} : \mathbb{R}^{I-1}_+ \to \mathbb{R}$ such that $\tilde{F}(d) = \sum_{k=1}^{I-1} d_k$. Problem (5) is equivalent to $\min_{d \in \mathcal{C} \cap \tilde{\mathcal{D}}} \tilde{F}(d)$. As a consequence of Proposition A.2, we can find a lower basic solution for system (15), with g as in (16) and h as in (18) with $\tilde{\mathcal{D}}$ in place of $\mathcal{D}$, by means of the iteration defined in Proposition 20 of [32].

Alternatively, we can use the Reverse Polyblock Approximation Algorithm of [34], based on a branch and bound procedure, which generates a sequence converging to a globally optimal solution of $\min_{d \in \mathcal{C} \cap \tilde{\mathcal{D}}} \tilde{F}(d)$. However, such algorithm suffers from some implementation issues discussed in [34] and its computational cost is very high.

As a different approach, we can use a pre-assigned set $\hat{\mathcal{X}}$ of large cardinality. That is, we consider a large set of possible initial positions, approximating all appropriate solutions with sufficient precision. This approach allows solving the problem directly by DP, avoiding the optimization of the x-coordinates of initial positions. However, this method can suffer from very high computational times and memory occupancy. Indeed, the number of explored states (and the computational time) increases very quickly with the cardinality of $\hat{\mathcal{X}}$.

REFERENCES

[1] H. Guo, Y. Wang, and W. Li, "Motion control technology of plc industrial palletizing robot," *Automation and Machine Learning*, vol. 3, no. 1, pp. 38–43, 2022.

[2] A. C. Caputo and P. M. Pelagagge, "Capacity upgrade criteria of large-intensive material handling and storage systems: a case study," *Journal of Manufacturing Technology Management*, vol. 19, no. 8, pp. 953–978, 2008.
[3] R. Chiba, T. Arai, T. Ueyama, T. Ogata, and J. Ota, "Working environment design for effective palletizing with a 6-dof manipulator," *International Journal of Advanced Robotic Systems*, vol. 13, no. 2, pp. 1–8, 2016.
[4] S. Lu, Y. Wu, and Y. Fu, "Research and design on pallet-throughout system based on rfid," in *Proceedings of the IEEE International Conference on Automation and Logistics*, August 2007, pp. 2592–2595.
[5] M. Laurini, L. Consolini, and M. Locatelli, "Optimizing cooperative pallet loading robots: A mixed integer approach," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5300–5307, July 2021.
[6] H. Dyckhoff, "A topology of cutting and packing problems," *European Journal of Operational Research*, vol. 44, pp. 145–159, 1990.
[7] C. S. Chen, S. M. Lee, and Q. S. Shen, "An analytical model for the container loading problem," *European Journal of Operational Research*, vol. 80, pp. 68–76, 1995.
[8] G. Fasano and J. D. Pintér, Eds., *Optimized Packings with Applications*. Springer, 2015, vol. 105.
[9] D. S. Johnson, "Fast algorithms for bin packing," *Journal of Computer and System Sciences*, vol. 8, no. 3, pp. 272–314, 1974.
[10] J. O. Berkey and P. Y. Wang, "Two-dimensional finite bin-packing algorithms," *Journal of the Operational Research Society*, vol. 38, no. 423–429, 1987.
[11] A. Soke and Z. Bingul, "Hybrid genetic algorithm and simulated annealing for two-dimensional non-guillotine rectangular packing problems," *Engineering Applications of Artificial Intelligence*, vol. 19, pp. 557–567, 2006.
[12] S. Martello, D. Pisinger, and D. Vigo, "The three-dimensional bin packing problem," *Operations Research*, vol. 48, no. 2, pp. 256–267, 2000.
[13] G. Abdou and E. Lee, "Contribution to the development of robotic palletization of multiple box sizes," *Journal of Manufacturing Systems*, vol. 11, no. 3, pp. 160–166, 1992.
[14] B. Ram, "The pallet loading problem: A survey," *International Journal of Production Economics*, vol. 28, pp. 217–225, 1992.
[15] S. Vargas-Osorio and C. Zúñiga, "A literature review on the pallet loading problem," *Lámpsakos*, no. 15, pp. 69–80, 2016.
[16] G. Scheithauer, *Introduction to Cutting and Packing Optimization Problems, Modeling Approaches, Solution Methods*, ser. Int. Series in Operations Research & Management Science. Springer, 2017.
[17] K. A. Dowsland, "An exact algorithm for the pallet loading problem," *European Journal of Operational Research*, vol. 31, pp. 78–84, 1987.
[18] ——, "Efficient automated pallet loading," *European Journal of Operational Research*, vol. 44, pp. 232–238, 1990.
[19] E. Silva, J. F. Oliveira, and G. Wäscher, "The pallet loading problem: a review of solution methods and computational experiments," *Int. Transactions in Operational Research*, vol. 23, pp. 147–172, 2016.
[20] J. Li and S. H. Masood, "Modelling robotic palletising process with two robots using queuing theory," *Journal of Achievements in Materials and Manufacturing Engineering*, vol. 31, no. 2, pp. 526–530, 2008.
[21] H. A. Khan and S. H. Masood, "Placement sequence methodology in pallet pattern formation in robotic palletisation," *Advanced Materials Research*, vol. 383–390, pp. 6347–6351, 2012.
[22] H. A. Khan, S. H. Masood, and A. Giecco, "An algorithm to determine placement sequence in robotic pallet pattern formation," *Advanced Materials Research*, vol. 403–408, pp. 3953–3958, 2012.
[23] S. H. Masood and H. A. Khan, "Development of pallet pattern placement strategies in robotic palletisation," *Assembly Automation*, vol. 34, no. 2, pp. 151–159, 2014.
[24] F. M. Moura and M. F. Silva, "Application for automatic programming of palletizing robots," in *18th IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, April 2018, pp. 48–53.
[25] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com
[26] I. I. Cplex, "V12. 1: User's manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.
[27] R. O. Saber, W. B. Dunbar, and R. M. Murray, "Cooperative control of multi-vehicle systems using cost graphs and optimization," in *Proceedings of the American Control Conference*, June 2003, pp. 2217–2222.
[28] A. Tsalatsanis, A. Yalcin, and K. P. Valavanis, "Optimized task allocation in cooperative robot teams," in *17th Mediterranean Conference on Control and Automation*, June 2009, pp. 270–275.
[29] L. E. Parker, "Decision making as optimization in multi-robot teams," in *Distributed Computing and Internet Technology*, R. Ramanujam and S. Ramaswamy, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 35–49.
[30] A. Baldassarri, G. Innero, R. D. Leva, G. Palli, and M. Carricato, "Development of a mobile robotized system for palletizing applications," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020, pp. 395–401.
[31] Y. He, M. Wu, and S. Liu, "A cooperative optimization strategy for distributed multi-robot manipulation with obstacle avoidance and internal performance maximization," *Mechatronics*, vol. 76, pp. 1–13, 2021.
[32] H. Tuy, "Normal sets, polyblocks, and monotonic optimization," *Vietnam Journal of Mathematics*, vol. 27, no. 4, pp. 277–300, 1999.
[33] J. K. Lenstra and A. H. G. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, pp. 221–227, 1981.
[34] H. Tuy, "Monotonic optimization: Problems and solution approaches," *SIAM Journal on Optimization*, vol. 11, no. 2, pp. 464–494, 2000.

Luca Consolini received the laurea degree (*cum laude*) in Electronic Engineering at the University of Parma in 2000, and the Ph.D. degree from the same university in 2005. From 2005 to 2009 he has been a postdoc at the University of Parma, where, from 2009 to 2014, he has been Assistant Professor. Since 2014, he is Associate Professor at the University of Parma. His main current research interests are nonlinear control, motion planning, and control of mechanical systems.

Mattia Laurini received the Bachelor's degree in 2013 and the Master's degree (*cum laude*) in 2015, both in Mathematics, at the University of Parma and the Ph.D. degree in Information Technologies at the same university. Currently, he is a Research Fellow at the Department of Engineering and Architecture of the University of Parma. His research interests lie in the areas of motion planning and optimization.

Marco Locatelli is Full Professor of Operations Research at the University of Parma. His main research interests are the theoretical, practical and applicative aspects of optimization. He published more than ninety papers in international journals and co-authored, with F. Schoen, the book *Global Optimization: Theory, Algorithms, and Applications* for the Society for Industrial and Applied Mathematics (SIAM). He has been nominated EUROPT Fellow in 2018. He is currently in the editorial board of the journals *Computational Optimization and Applications*, *Journal of Global Optimization*, and *Operations Research Forum*.