Optimizing Container Loading with Autonomous Robots

Demetris Stavrou*, *Member, IEEE,* Stelios Timotheou†, *Senior Member, IEEE,* Christos G. Panayiotou†, *Senior Member, IEEE,* and Marios M. Polycarpou†, *Fellow Member, IEEE*

***Abstract*—In this work, we investigate a problem associated with transferring a set of containers from the storage to the loading area of a warehouse using autonomous robots. In addition to assigning robots to containers, the special topology considered in this work requires coordinated planning of the robots' movement to avoid conflicts. We formulate the joint problem of robot assignment and movement coordination with the objective of minimizing the time required for all robots to carry their assigned containers to the destination, subject to conflict-free movement of all robots. We use the concept of Abstract Time-Windows to represent the movement of robots. The conditions for detecting conflicts in the Abstract Time-Window representation are introduced along with the necessary operations for resolving conflicts. For the solution of the problem, two approaches are developed. The first, is a mathematical programming approach that formulates the problem as a Mixed Integer Linear Program that allows optimal solution using appropriate solvers, while the second, is a heuristic approach that allows fast, close-to-optimal solutions. Even though the proposed approaches focus on the case where the number of robots is equal to the number of containers, we also discuss how to solve the problem when having unequal number of robots and containers. Simulation results show that the heuristic approach provides a solution within 5% of the optimal solution with the minimum time of completion of all tasks being the performance metric, and executes six orders of magnitude faster than a state-of-the-art mathematical programming solver.**

Note to Practitioners: ***Abstract*—Robotic systems are increasingly used in logistics facilities such as warehouses and container terminals. Nonetheless, achieving improved efficiency in this context poses several challenges in terms of real-time execution, coordinated resolution of robot conflicts and adaptation to unpredictable events in operational environments. This work examines a problem related to the coordination of a team of autonomous robots operating in a container handling facility, where containers need to be transferred from the storage area to a loading station. The considered topology involves containers arranged in lanes for efficient storage, but imposes limitations on the movement of autonomous robots. Such a topology is commonly found in container terminals where straddle carriers have to transfer containers from a storage yard to quay cranes or trucks and vice-versa, but have limitations on moving on the same or even adjacent container lanes. To address this problem, we develop a novel method that efficiently assigns containers to robots and defines appropriate timings for collective coordination of robot movements. We illustrate that the proposed method provides very efficient results (within 5% from the optimal) for multiple configuration scenarios. Even more importantly, the method is suitable for real-time execution as it solves problems with hundreds of robots/containers within a few milliseconds; this further offers the ability for solution adaptation in dynamically varying environments, e.g. in situations of robot delays or breakdowns.**

***Index Terms*—Autonomous robots, multi-robot assignment and coordination, container handling facility, optimization.**

I. Introduction

AUTOMATION is a key factor for improving efficiency in logistics applications. For this reason, robotic systems are extensively used in various facilities such as warehouses, container terminals and transportation systems [1]–[4]. A facility as such, consists of multiple autonomous mobile robots used as carriers for transporting containers from storage areas to handling stations by following physically or virtually defined paths.

To reduce the storage costs, operators have an incentive to increase the number of containers that are stored in the facility, which limits the available space left for the robots to travel into. Also, increasing the number of robots operating simultaneously, increases the throughput of the facility. However, as the number of robots increases and the available travel space decreases, contention becomes greater issue. For example, a conflict situation can occur when two robots travel towards each other in opposite directions, in a path wide enough to fit only one of them. In this context, we examine a problem requiring the assignment and movement coordination of multiple robots in a container warehouse to minimize the maximum time needed to complete all tasks.

Due to similarities, one may relate this problem to the *Vehicle Routing Problem* (**VRP**) [5] which considers a fleet of vehicles, all with the same capabilities, used to deliver goods to a set of customers located at different locations and then return back to the depot. This problem has received great attention by many researchers and many solutions have been proposed for the VRP [6] as well as its variants such as: Vehicle Routing Problem with Time Windows, Capacitated Vehicle Routing Problem and Vehicle Routing Problem with Pick-Up and Delivery. However, there are significant differences with the problem considered in this work. The spatial scale of the VRP is much broader than the considered problem, since nodes in the network usually represent locations within a city. This draws the attention of the problem away from collisions or congestion that may occur when vehicles are operating. In a facility with limited free space such as a

∗ D. Stavrou is with Phoebe Innovations, Nicosia, Cyprus (email: stavrou.demetris@ucy.ac.cy)

† S. Timotheou, C. G. Panayiotou and M. M. Polycarpou are with KIOS Research Center for Intelligent Systems and Networks, and the Dept. Electrical and Computer Engineering, University of Cyprus. (emails: {timotheou.stelios, christosp, mpolycar}@ucy.ac.cy)

This work is partially funded by the European Research Council Advanced Grant FAULT-ADAPTIVE (ERC-2011-ADG-291508).

warehouse, conflicts and congestion have to be explicitly taken into account. Furthermore, in the VRP case the shortest path normally generates the fastest time but in a facility with multiple vehicles this is not the case. These significant characteristics motivate us to treat this problem differently.

Automating a facility with mobile robots involves solving the scheduling and routing problems [7]. The purpose of scheduling is to coordinate the available vehicles by designating which container should be handled by each vehicle, usually under the constraint of priority and with the objective of minimizing the time. Given successful scheduling, the purpose of routing is to discover an efficient path, with respect to time, between each vehicle and it's destination. The algorithms must ensure that the vehicles reach their destination.

One way to approach this problem is through the design of the path network on which the autonomous robots move between pick-up and delivery stations prior to the setup of the facility. By focusing on the path network design, more efficient routing solutions can be obtained at the expense of less flexibility in the configuration of the facility [8], [9]. A second approach decomposes the facility into non-overlapping, single vehicle loops that operate in tandem, in a way that the exchange of containers is achieved through transfer stations positioned between adjacent loops [10]–[12]. This implies that each container is handled by more than one vehicles before reaching its destination, introducing significant overhead due to the multiple loading/unloading procedures. A third approach is to segment the path network into logical zones and then impose rules on each zone to predict and avoid deadlocks, e.g. restricting the number of vehicles allowed within a specific zone [13], [14] or employing a Petri-net formalism to model and control deadlocks [15], [16]. Zones can result in sub-optimal utilization of spatial resources, as zones can only be occupied by one vehicle at a time. Reducing the zone size down to vehicle size addresses this issue, but increases the computational complexity of the routing. A fourth approach jointly considers the design of the facility and the path network, as well as the routing of the robots to maximize the performance of the overall system [17].

In cases that the movement of the vehicles is not restricted, i.e. they are able to move anywhere within the facility, vehicle paths can be calculated a-priori or on-line. Conflicts can be avoided by extensive planning and utilizing the spatial resources [18] or both spatial and temporal resources of the path network [19]–[22]. Such methods generate efficient solutions but the solution space expands rapidly as the number of network paths and the number of vehicles increases, requiring significant computational power, making them unsuitable for real-time execution. To deal with real-time constraints, a common practice is to calculate these paths a-priori. However, unexpected events such as delayed movement, running ahead of schedule or temporary hardware malfunction, may lead to conflicts which invalidate the precomputed paths, hence requiring a new solution. In an effort to lower the complexity, researchers describe a conservative myopic strategy i.e. vehicles are routed one-by-one while all the previous route decisions are respected [23].

In this work, we consider a problem associated with autonomous robots loading and delivering containers located in a warehouse facility. In terms of scheduling, an *assignment problem* needs to be solved indicating which robot should undertake each task. In terms of routing, a *coordination problem* needs to be addressed describing how the robots should coordinate their movements in order to avoid conflicts and reach their destination. The topology considered in this work has specific characteristics in terms of container arrangement, loading/unloading procedures and movement constraints. These characteristics introduce a number of challenges that distinguish the considered problem from problems addressing assignment and coordination of autonomous robots in relevant applications. First, the facility topology does not allow alternative paths for the loading and delivery of each container. In fact, a large percentage of containers may have conflicting path segments which makes conflict resolution highly complex. This requires very careful a-priori coordinated planning of the robot paths to avoid conflicts, contrary to most methods that attempt to resolve conflicts on-the-fly with the risk of ultimately not reaching a conflict-free solution. Second, achieving an efficient solution in this confined environment poses a significant challenge; online rerouting in such a highly confined environment is expected to introduce further rerouting causing serious inefficiencies to the facility [24]–[26]. The same issue extends to the assignment of containers to robots, as many approaches employ simple algorithms such as first-come-first-served [7]. Third, solutions provided by high-complexity a-priori approaches are not desirable in our case, because the dynamic nature of such facilities requires real-time decision making.

The contributions of this work are the following:

- Formulation and optimal solution of the assignment and coordination problem using Mixed Integer Linear Programming (MILP) tools
- Development of a low time-complexity polynomial algorithm for solving the assignment and coordination problem that provides close to optimal results
- Description of a reduced time-complexity algorithm for a special container arrangement
- Outline of a heuristic rolling horizon methodology for the case where multiple tasks need to be executed by each robot
- Theoretical analysis of the computational performance of the heuristic algorithm with respect to the optimal solution and other conflict resolution strategies

Each of the two developed approaches serves an important role. On the one hand, the MILP approach provides optimal performance, is suitable for the solution of problems where task execution requests arrive ahead of time, and serves as a performance benchmark against other developed algorithms. On the other hand, the heuristic approach provides close-to-optimal performance, has low computational complexity, and is suitable for the solution of problems where requests arrive at the last minute or unexpected events take place such as robot delays and breakdowns. Although both approaches are developed for the case where the number of containers is equal to the number of robots, it is also described how the heuristic

algorithm can be modified to solve the problem when having unequal number of robots and containers.

Section II describes the problem formulation, including the objective and constraints. Section III explains how the movement of robots can be transformed into abstract time-windows that allow conflict detection and resolution, and also defines the conditions that provide conflict-free movement for all robots. Section IV formulates and solves the problem using Mixed-Integer Linear Programming (MILP), while Section V develops the proposed low time-complexity polynomial algorithm for the solution of the problem. Section VI investigates the performance of the proposed heuristic algorithm compared to the optimal solution in terms of solution quality and execution speed. Finally, Section VII concludes the paper.

Notation: We use the upper case boldface letters for matrices, lower case boldface letters for vectors and calligraphic letters for sets. $(\cdot)^\top$ denotes the transpose of a matrix or vector. $|\mathcal{S}|$ denotes the cardinallity of set $\mathcal{S}$, while the i-th element of set $\mathcal{S}$ is denoted by $\mathcal{S}(i)$. Operators $\wedge$, $\vee$, $\oplus$ denote the logical functions AND, OR and XOR.

II. Problem Statement

The motivation of this work emanates from a common problem encountered in container terminals where containers are stacked into lanes: straddle carriers that handle the containers cannot move next to each other on adjacent lanes, as the space between lanes is enough only for one straddle carrier, as depicted in Fig. 1. This space limitation creates the challenge of simultaneously loading multiple containers without any movement conflicts between straddle carriers.

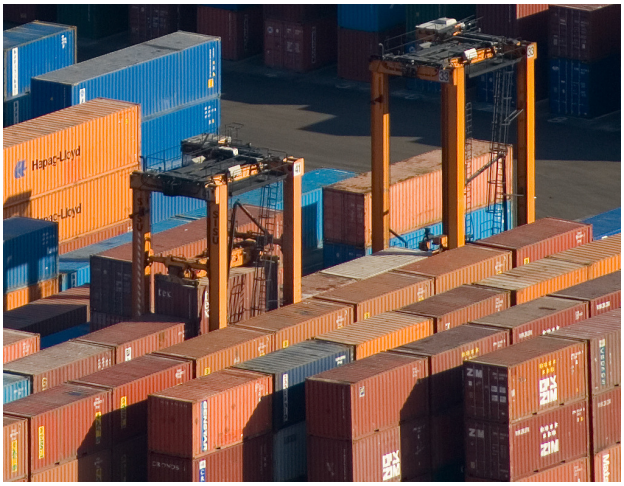


Fig. 1. Straddle carriers operating in container terminal. (Used with permission by the owner, *Sergio Morchon*).

In this work, we consider a topology inspired from the one used in container terminals as illustrated in Fig. 2. In general, container flow involves two interfaces: the quayside where ships are loaded and unloaded and the land-side where containers are loaded and unloaded to/from trucks and trains. The illustrated area in Fig. 2 is a representation of the straddle carrier container stack which is the buffer between the quayside and land-side interfaces [2], [27]. The area is divided in two distinct regions, the storage region where all containers are stored and the free-moving region which is the shaded region in the figure. Containers need to be transported from the storage area to the loading area. A robot, similar to the straddle carrier, is equipped with wheels located along its two sides. In order to load a container the robot needs to be positioned above it, placing its wheels to its two sides and finally lifts it up. The robot is tall enough such that when carrying a container, there is enough clearance underneath to move over other containers without hitting them. To save space, containers are stacked into long lanes, and the limited space between two consecutive lanes forms the transportation aisles of the warehouse. The free-moving region has no constrained paths therefore a robot can move freely in any direction. In this region, robots can maneuver and avoid collisions, assuming that the robots are equipped with obstacle detection sensors. Given the approximate positions of the robots, appropriate paths can be calculated to prevent any potential collisions between them. This however, is outside the scope of this work, therefore we rely on existing algorithms to resolve such issues [28]–[30].

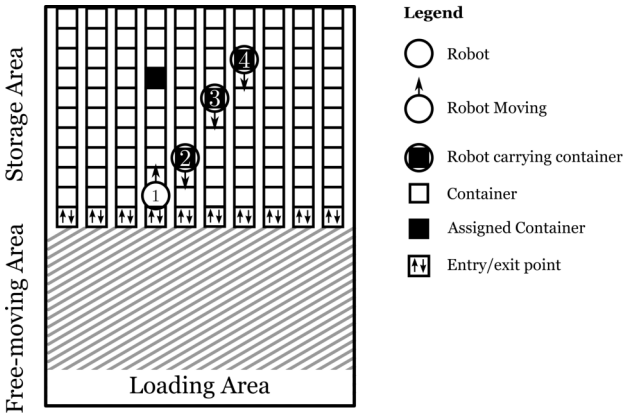


Fig. 2. The specific facility topology considered in this work. The limited space between the container lanes can cause conflict between two opposite moving robots, as illustrated in this figure with robots 1 and 2.

The facility is equipped with a set $\mathcal{R}$, $|\mathcal{R}| = n$, of interchangeable robots for transporting the containers. We define *task* $s \in \mathcal{S}$ as the process during which robot transports container s from its location in the storage area to the loading area, with $\mathcal{S}$, $|\mathcal{S}| = n$, being the set of tasks that the operator of the facility requests to be transported and loaded. Requests arrive either ahead of time or at the last-minute. Hence, while there is enough time to find the best solution for transporting the containers in the first case, the second case requires immediate computation of an effective solution. Notice that for simplicity we have assumed that the number of robots is equal to the number of containers. The case $|\mathcal{S}| \neq |\mathcal{R}|$ is discussed in Section V-B.

In terms of storage space, this topology is efficient. However, due to the extremely limited space in the container storage region, special constraints exist that distinguish this problem from other general topology problems. The long container lanes have a single entry/exit point at the lower end. This means there is only one way to reach any given container and also it is not possible to switch lanes once a robot enters

a lane. In order for a robot to switch a lane, it has to travel all the way to the exit of the current lane, move to the entrance of the desired lane and then move up to the desired container. Therefore, even two spatially close containers in different lanes are actually very far in terms of robot path.

Another constraint that arises due to the limited resources, is the movement of two or more robots in adjacent lanes. The aisles are wide enough for only one robot to use, so it is not possible for two robots to be on adjacent lanes, i.e. side-by-side. An example of this situation is shown in Fig. 2. Robot 1 is moving upwards while robot 2 is moving downwards. When they meet they enter into a conflict because each one opposes the progress of the other. Later on, robots 3 and 4 which also travel downwards will join the conflict, and this shows how fast it can escalate in the facility.

A. Problem Decomposition

Addressing our problem, requires the consideration of two related problems: assignment and coordination. Even though the two problems could be treated separately, joint consideration of these provides better results.

Regarding the assignment problem, the binary matrix $\mathbf{X} \in \{0,1\}^{n\times n}$ denotes the assignments of robots to tasks with element $x_{i,s}$ being equal to 1 if task s has been assigned to robot i and 0 otherwise. In addition, each robot should be assigned only one task and no two robots should be assigned to the same task, and every task should be assigned to only one robot i.e.:

$$\sum_{i=1}^{n} x_{i,s} = 1,\ s \in \mathcal{S} \text{ and } \sum_{s=1}^{n} x_{i,s} = 1, i \in \mathcal{R}.$$

Due to the constraints of the problem, a robot assigned to task s in lane $l_s \in \mathcal{L}$ may conflict with robots assigned to tasks on the same or directly adjacent lanes, i.e. $l_s - 1$, l_s and $l_s + 1$. We define set $\mathcal{C}_i$ as the set of robots which may conflict with robot i, i.e. those located on directly adjacent or on the same lane with robot i. Because it is possible to have many robots moving on each lane, conflicts may propagate across the entire facility, so that conflict resolution has to be considered simultaneously for all lanes. The robots have to coordinate with respect to their entrance/exit order in different lanes as well as their movement strategy (when to move and when to stand still). Initially, each robot may have to wait for a certain period of time before moving towards its defined lane in order to respect the decided entrance/exit order. During this period, the robot will remain within the free-moving area so that other robots are able to maneuver around it, avoiding a possible collision. It is also possible for a robot to wait at the location of its designated task for another robot to exit before exiting the specific lane.

B. Objective Function

After being assigned a task, each robot starts to move towards the entrance of the lane the container is in. It proceeds to move and pickup the container, then exits the lane and finally delivers the container at the loading area. The time required for robot i to finish its complete movement is defined as:

$$T_i(\mathbf{x}_i, w_i^e, w_i^x) = \sum_{s\in\mathcal{S}} T_{i,s}^e x_{i,s} + w_i^e + T^l + 2\sum_{s\in\mathcal{S}} T_s^v x_{i,s} + w_i^x + T^d, \tag{1}$$

where $\mathbf{x}_i$ is the assignment vector for robot i, i.e. $x_{i,s} = 1$ if robot i is assigned to task s and 0 otherwise, $T_{i,s}^e \in \mathbb{R}^+$ is the travel time to the lane entrance where task s is located, $w_i^e \in \mathbb{R}^+$ is the *entrance waiting time*, i.e., the time before robot i begins its movement from its initial position in the free-moving area, $T_s^v \in \mathbb{R}^+$ is the travel time from the lane entrance to the s-th container's position, and $w_i^x \in \mathbb{R}^+$ is the *exit waiting time*, i.e., the waiting time at the location of the container before the robot travels towards the lane exit. The objective function is subject to the conflict-free condition which is described in detail in Section III. All robots require the same time to load a container which is defined as $T^l \in \mathbb{R}^+$. In addition, $T^d \in \mathbb{R}^+$ is the travel time from each lane exit to the loading area. Note that when robots enter and exit the lanes, they should keep a time-distance apart defined as guard time $T^g \in \mathbb{R}^+$ for safety reasons. Guard time also treats potential small delays imposed by obstacle avoidance maneuvers in the free-moving area.

There are different metrics for performance such as *minimum makespan*, *maximum throughput*, *minimum travel* and *even distribution of workload* [31]. In this work we consider the *minimum makespan* as the objective of interest which is equivalent to the minimization of the time at which the last task is completed or equivalently the minimization of the maximum time required by any robot to complete its task, i.e.:

$$\Lambda(\mathbf{X}, \mathbf{w}^e, \mathbf{w}^x) = \max_{i\in\mathcal{R}} T_i(\mathbf{x}_i, w_i^e, w_i^x) \tag{2}$$

This equation is subject to having no conflicts between any of the robots. Under certain conditions, robot i may conflict with one or more robots defined as the set $\mathcal{C}_i$. The objective is to find a set of values $\{\mathbf{X}, \mathbf{w}^e, \mathbf{w}^x\}$ such that

$$\{\mathbf{X}^*, \mathbf{w}^{e*}, \mathbf{w}^{x*}\} = \underset{\{\mathbf{X}, \mathbf{w}^e, \mathbf{w}^x\}}{\operatorname{argmin}} \Lambda(\mathbf{X}, \mathbf{w}^e, \mathbf{w}^x) \tag{3}$$

In order to derive the optimal values for $\mathbf{X}, \mathbf{w}^e$ and $\mathbf{w}^x$ therefore, an optimization problem has to be solved. Note that time T^d increases the completion time of all tasks equally, but does not affect the specification of the optimal solution as the assignment and waiting times remain the same irrespective of the presence of T^d in the problem. The reason is that the particular parameter is associated with traveling in the free-moving area at the end of movement of each robot so that no conflict is created as a result of this parameter.

C. Illustrative Example

To gain a better insight into the problem, in this paragraph we investigate three different cases that can arise in a simple configuration. We assume a small facility composed of only one lane of containers, two robots and two tasks, and assume that each robot moves 1 grid cell per iteration. The initial configuration at iteration $\tau = 0$ is shown in Fig. 3(a). In this example, s_1 is assigned to robot 1 and s_2 to robot 2. For simplicity we consider $T^l = 0$.

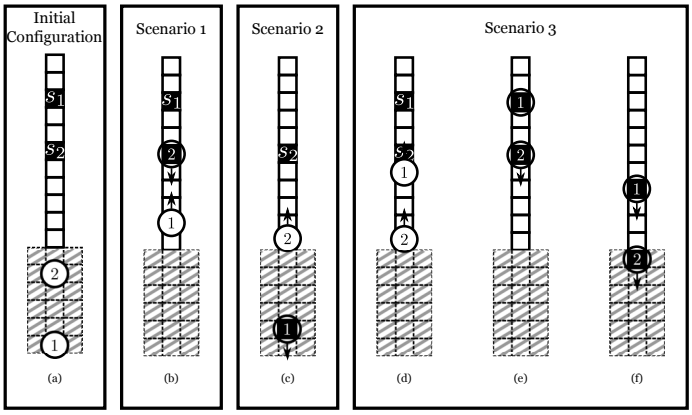


Fig. 3. Illustrative example with 2 robots, 2 containers and 1 lane. (a) The initial configuration. (b) Scenario 1 (c) Scenario 2 (d)(e)(f) Scenario 3

- First we consider the case where both robots start moving towards their assigned tasks without any waiting times, i.e. $\mathbf{w}^e = [0,0]^\top$ and $\mathbf{w}^x = [0,0]^\top$. At $\tau = 7$, robot 2 reaches its task, loads the container and is ready to start moving towards the exit. Robot 1 on the other hand is still traveling towards its container, as shown in Figure 3(b). Inevitably, the robots will conflict at $\tau = 9$.
- Let us consider a second case where the waiting times are adjusted as $\mathbf{w}^e = [0,25]^\top$ in order to prevent conflicts and have the appropriate guard time $T^g = 4$. Robot 1 starts moving first towards its task while robot 2 waits. Robot 1 reaches its task at $\tau = 14$ and exits the lane at $\tau = 23$. Robot 2 will only start moving at $\tau = 25$ and enters the lane at $\tau = 27$ as shown in Fig. 3(c). The time between robot 1 exiting and robot 2 entering is enough to satisfy the guard time constraint. Finally, robot 2 will exit at $\tau = 38$.
- In the third case, we solve the same problem differently, using $\mathbf{w}^e = [0,8]^\top$ and $\mathbf{w}^x = [2,0]^\top$. Robot 1 starts moving first, while robot 2 waits. At $\tau = 6$ robot 1 enters the lane and at $\tau = 9$ robot 2 starts moving as well so it enters the lane at $\tau = 10$ so they respect T^g. This is shown in Fig. 3(d). After robot 1 loads its container, it has to wait until robot 2 reaches its task at $\tau = 16$ (shown in Fig. 3(e)) and start moving at $\tau = 17$. This way robot 2 exits at $\tau = 21$, shown in Fig. 3(f), and robot 1 at $\tau = 25$ and therefore respect the guard time.

The above example demonstrates the solution of the problem using different sets of waiting times. Apart from the considered assignment, there is also the option for s_1 assigned to robot 2 and s_2 assigned to robot 1. The assignment changes the initial conditions of the problem significantly affecting the solution. For exploring the full set of solutions, one would need to consider all possible assignment combinations. However, this example is meant to demonstrate the impact of waiting times in eq. (1) on the solution of the problem, therefore only one assignment combination is considered. Also note that the use of time and space discretization in this example is for illustration purposes; both the MILP and approaches developed in Sections IV and V, employ continuous time and space.

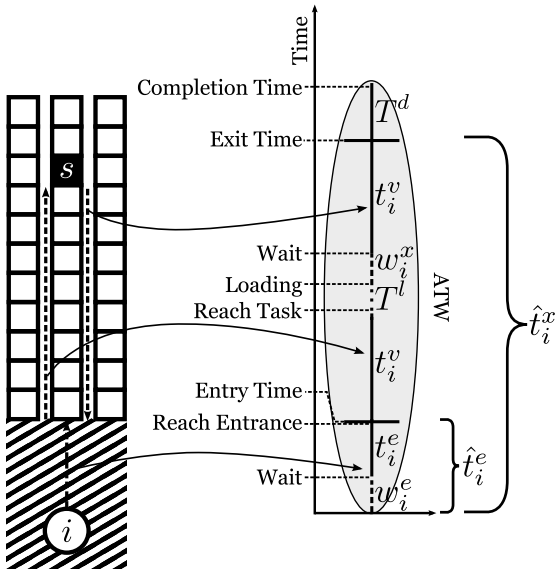


Fig. 4. Schematic representation of robot movement in the container lane topology. The Abstract Time-Windows (ATW) transformation provides a tool for studying the problem, detecting conflicts between robots and for resolving conflicts with the allowed ATW operations.

III. Abstract Time-Windows

In this section we describe how the time-line of a robot's movement can be graphically represented, inspired by the concept of *Time Windows* used in Operational Research. This representation is called *Abstract Time Windows* (ATW) and provides a method to study the movement of all robots collectively, detect conflicts and resolve them. At this point, for notational clarity we define $t_i^e = \sum_{s\in\mathcal{S}} T_{i,s}^e x_{i,s}$ and $\hat{t}_i^e = t_i^e + w_i^e$ as the time required for robot i to reach the storage region without/with initial waiting time, respectively. Also $t_i^v = \sum_{s\in\mathcal{S}} T_s^v x_{i,s}$ is the travel time that robot i requires to move from the entrance of the storage region to its assigned container. Finally, we define $t_i^x = 2t_i^v + t_i^e + T^l$ and $\hat{t}_i^x = t_i^x + w_i^e + w_i^x$, which denote the exit time of robot i from the container area without and with waiting.

An ATW represents the time-line of the movement of a specific robot and indicates all traveling and waiting times. An ATW is schematically represented by a line with distinct start and end parts as depicted in Fig. 4. Initially, an ATW describes the best-case scenario of a robot moving from its starting position to the container location and then exiting the lane. Therefore, given an assignment, an ATW starts at time t^e then extends by $2t^v$ depending on the location of the container and finally ends. If required, an ATW can be modified to include waiting times w^e and w^x. The ATW ends at $\hat{t}^x + T^d$.

In a facility where $n > 1$, ATWs are used to detect conflicts using the following condition.

Conflict-Free Condition (CFC): Consider any pair of conflicting robots i and k such that $t_i^e \leq t_k^e$ and $k \in \mathcal{C}_i$. Then, these robots can navigate with no conflicts with respect to each other, if exactly one of the following two cases is true:

1) $\hat{t}_i^x + T^g < \hat{t}_k^e$
2) $(\hat{t}_i^e + T^g < \hat{t}_k^e) \wedge (\hat{t}_i^x + T^g > \hat{t}_k^x)$

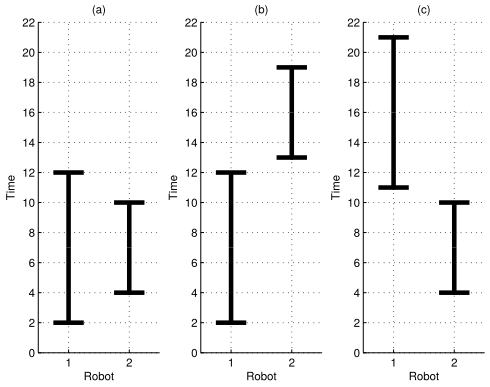


Fig. 5. There are only 3 possible configurations between two ATWs that satisfy the CFC.

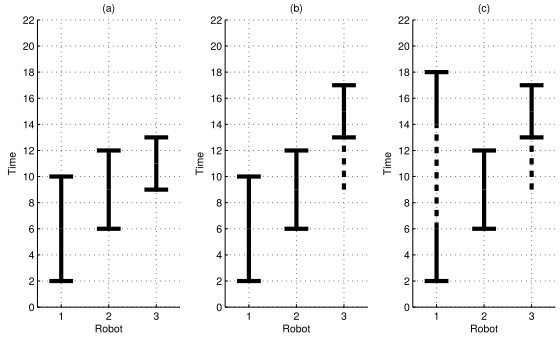


Fig. 6. An example illustrating conflict resolution using ATWs.

A solution satisfies the Conflict-Free Condition if all pairwise conflicting robot combinations satisfy the above condition.

This holds because if robot i enters the lane before robot k then there are only two possible outcomes in order to progress without conflicts. The first is when robot i exits safely before robot k enters and the second is when robot k safely enters after robot i and exits before robot i exits. In general, when having ATWs i and k then there are only 3 possible configurations between them that satisfy the CFC which are shown schematically in Fig. 5.

If CFC is violated then one can use the following two operations in order to resolve the conflict and make the solution CFC viable:

Shift: Shift ATW by increasing its w^e value.

Extend: Extend ATW by increasing its w^x value.

In other words, the ATW can be shifted upwards or it can be extended but it cannot be compressed. A shift upwards implies that the robot needs to wait before starting to move for w^e time units, while extension implies that the robot needs to wait at the task location for w^x time units. Fig. 6(a) shows an example with three ATWs where the CFC is violated in two cases. ATW 2 conflicts with ATW 3 and also ATW 1 conflicts with both ATW 2 and 3. To resolve the first conflict we can use operation Shift and shift ATW 3 as shown in Fig. 6(b), i.e. robot 3 will need to wait before starting its movement so that it enters the storage section after robot 2 has exited. To resolve the second conflict, ATW 1 is extended using operation Extend as shown in Fig. 6(c). The solution satisfies CFC.

IV. MILP Formulation

The optimal solution of the considered problem is achieved by incorporating the derived CFCs into a Mixed-Integer Linear Programming (MILP) formulation. MILP formulations for achieving deadlock-free operation have also been considered in various configurations with different requirements such as for job-shop scheduling with a stationary robot distributing jobs to a set of machines [32]. To achieve this, certain logical constraints (LC1–LC4) need to be transformed into equivalent MILP constraints as shown in Table I[1]. In the table, constants M_k^u (M_k^l) is an upper (lower) bound on $\sum_i x_i a_{i,k} - b_k$, while ϵ is a small tolerance beyond which the associated constraint is not true. For instance, the logical constraint LC3 which indicates that "if $z = 1$ then exactly one constraint (XOR ($\oplus$) operation) between $\sum_i x_i a_{i,1} \leq b_1$ and $\sum_i x_i a_{i,2} \leq b_2$ is true", can be equivalently expressed from the two associates inequalities where variable δ indicates whether the first ($\delta = 1$) or the second ($\delta = 0$) constraint is true.

The MILP formulation of the considered problem is based on the fact that the assignment matrix $\mathbf{X}$, and the waiting vectors $\mathbf{w}^e$ and $\mathbf{w}^x$ must be optimally selected in order to minimize the total cost, defined as $f_T = \max_{i\in\mathcal{R}} \hat{t}_i^x$, and at the same time ensure that there is no conflict between any pair of robots. Towards this direction, the approach taken is to define appropriate MILP constraints based on the equivalent representation of LC1–LC4 with MILP constraints ensuring that whenever two robots are potentially conflicting, i.e. they have been assigned tasks in the same or neighboring lanes, exactly one of the two CFC cases holds true. The developed MILP formulation for the optimal solution of the considered problem is given below:

$$\min_{\{\mathbf{X},\ \mathbf{w}^e,\ \mathbf{w}^x,\ \zeta,\ \hat{\mathbf{t}}^x,\ \hat{\mathbf{t}}^e,\ \boldsymbol{\delta},\ \hat{\boldsymbol{\delta}},\ \tilde{\boldsymbol{\delta}},\ \boldsymbol{\psi},\ \boldsymbol{\xi}\}} \quad \zeta \tag{4a}$$

$$\text{s.t. } \hat{t}_i^x \leq \zeta,\ i \in \mathcal{R}, \tag{4b}$$

$$\hat{t}_i^x = 2\sum_{s\in\mathcal{S}} T_s^v x_{i,s} + \hat{t}_i^e + T^l + w_i^x,\ i \in \mathcal{R}, \tag{4c}$$

$$\hat{t}_i^e = \sum_{s\in\mathcal{S}} T_{i,s}^e x_{i,s} + w_i^e,\ i \in \mathcal{R}, \tag{4d}$$

$$\sum_{s\in\mathcal{S}} x_{i,s} = 1,\ i \in \mathcal{R}, \tag{4e}$$

$$\sum_{i\in\mathcal{R}} x_{i,s} = 1,\ s \in \mathcal{S}, \tag{4f}$$

$$\sum_{s\in\mathcal{S}} x_{i,s} l_s - \sum_{s\in\mathcal{S}} x_{k,s} l_s - (M_2^l - \epsilon)\hat{\delta}_{i,k} \geq 1 + \epsilon, \quad i,k \in \mathcal{R},\ k \neq i, \tag{4g}$$

$$-\sum_{s\in\mathcal{S}} x_{i,s} l_s + \sum_{s\in\mathcal{S}} x_{k,s} l_s - (M_2^l - \epsilon)\tilde{\delta}_{i,k} \geq 1 + \epsilon, \quad i,k \in \mathcal{R},\ k \neq i, \tag{4h}$$

$$\delta_{i,k} \geq \hat{\delta}_{i,k} + \tilde{\delta}_{i,k} - 1,\ i,k \in \mathcal{R},\ k \neq i, \tag{4i}$$

$$\hat{t}_i^e - \hat{t}_k^e \leq M_1^u(1 - \psi_{i,k}) - T^g,\ i,k \in \mathcal{R},\ k \neq i, \tag{4j}$$

$$\psi_{i,k} + \psi_{k,i} = \delta_{i,k},\ i,k \in \mathcal{R},\ k < i, \tag{4k}$$

[1]For details about the derivation of the equivalence between these logical and MILP constraints see [33].

ID	Logical Constraint	Equivalent MILP Expressions
LC1	$z=1 \rightarrow \sum_i x_i a_i \leq b$	$\sum_i x_i a_i - b \leq M^u(1-z)$
LC2	$z=1 \rightarrow (\sum_i x_i a_{i,1} \leq b_1) \oplus$ $\oplus\ (\sum_i x_i a_{i,2} \leq b_2)$	$\sum_i x_i a_{i,1} - b_1 \leq M_1^u(1-z) + M_1^u(1-\delta)$ $\sum_i x_i a_{i,2} - b_2 \leq M_2^u(1-z) + M_2^u\delta,\ \delta \in \{0,1\}$
LC3	$z=1 \rightarrow (\sum_i x_i a_{i,1} \leq b_1) \wedge ...$ $\wedge\ (\sum_i x_i a_{i,k} \leq b_k),\ k=1,...,K$	$\sum_i x_i a_{i,k} - b_k \leq M_k^u(1-z),\ k=1,...,K.$
LC4	$(\sum_i x_i a_{i,1} \leq b_1) \wedge$ $\wedge\ (\sum_i x_i a_{i,2} \leq b_2) \rightarrow z=1$	$\sum_i x_i a_{i,k} - b_k \geq (M_k^l - \epsilon)\delta_k + \epsilon,\ k=1,2.$ $z \geq \delta_1 + \delta_2 - 1, \quad z,\ \delta_1,\ \delta_2 \in \{0,1\}$

TABLE I
EQUIVALENT MILP EXPRESSIONS OF SPECIFIC LOGICAL CONSTRAINTS.

$$\hat{t}_i^x - \hat{t}_k^e \leq -T^g + M_1^u(1-\psi_{i,k}) + M_1^u(1-\xi_{i,k}), \quad i,k \in \mathcal{R},\ k \neq i, \quad (4l)$$

$$\sum_{s\in\mathcal{S}} T_s^v x_{k,s} - \sum_{s\in\mathcal{S}} T_s^v x_{i,s} \leq M_1^u(1-\psi_{i,k}) + M_1^u\xi_{i,k}, \quad i,k \in \mathcal{R},\ k \neq i, \quad (4m)$$

$$\hat{t}_k^x - \hat{t}_i^x \leq -T^g + M_1^u(1-\psi_{i,k}) + M_1^u\xi_{i,k}, \quad i,k \in \mathcal{R},\ k \neq i, \quad (4n)$$

$$w_i^e \geq 0,\ w_i^x \geq 0,\ i \in \mathcal{R}, \quad (4o)$$

$$\delta_{i,k}, \hat{\delta}_{i,k}, \tilde{\delta}_{i,k}, \psi_{i,k}, \xi_{i,k} \in \{0,1\},\ i,k \in \mathcal{R}, k \neq i, \quad (4p)$$

$$x_{i,s} \in \{0,1\},\ i \in \mathcal{R},\ s \in \mathcal{S}. \quad (4q)$$

In formulation (4) constants M_2^l and M_1^u denote lower and upper bounds of quantities $x_{i,s}l_s - \sum_{s\in\mathcal{S}} x_{k,s}l_s - 1$ and $\hat{t}_k^x - \hat{t}_i^x + T^g$, respectively. In particular, $M_2^l = -\max_{s\in\mathcal{S}}\{l_s\} - 2$, while M_1^u is obtained by finding a lower bound for $\hat{t}_i^x = w_i^x + 2t_i^v + T^l + t_i^e + w_i^e$ and an upper bound for $\hat{t}_k^x = w_k^x + 2t_k^v + T^l + t_k^e + w_k^e$. In particular, a lower bound for $\hat{t}_i^x$ is obtained by setting the two waiting times equal to zero and selecting the minimum value of $2t_i^v + t_i^e$; this yields $\hat{t}_i^x \geq 2T_{\check{s}}^v + T_{\check{i},\check{s}}^e + T^l$, where $\{\check{i},\check{s}\} = \text{argmin}_{i\in\mathcal{R},s\in\mathcal{S}}\{2T_s^v + T_{i,s}^e\}$. An upper bound for $\hat{t}_k^x$ corresponds to the objective value of a feasible solution. One such solution can be constructed using a simple strategy where robot k enters into the loading area before and exists after robot i if $t_k^v > t_i^v$. In this case, it is important to impose enough entrance and exit waiting times to robot k, w_k^e and w_k^x, such that the associated abstract time-window (ATW) embeds all other ATWs i with $t_i^v < t_k^v$. Let $s^m = \text{argmax}_{s\in\mathcal{S}}\{T_s^v\}$ denote the task with the largest T_s^v value and i_m the robot that executes the particular task. In this setting, it is true that i^m needs to have enough entrance waiting time such that $w_{i^m}^e + t_{i^m}^e \geq t_i^e + T^g$, $i \neq i^m$. This is true when $w_{i^m}^e \leq \hat{W}^e = max_{i\in\mathcal{R},s\in\mathcal{S}} T_{i,s}^e + (n-1)T^g$. In term of exit waiting time, it is true that $w_{i^m}^x \leq \hat{W}^x = (n-1)T^g$ as the particular ATW needs to have at least T^g gap above all other ATWs. Hence, it is true that $M_1^u \geq \hat{t}_k^x - \hat{t}_i^x + T^g$ when:

$$\begin{aligned} M_1^u &= \hat{W}^e + \hat{W}^x + 2T_{s^m}^v - (2T_{\check{s}}^v + T_{\check{i},\check{s}}^e) + T^g \\ &= \max_{i\in\mathcal{R},s\in\mathcal{S}}\{T_{i,s}^e\} + (2n-1)T^g + 2T_{s^m}^v - (2T_{\check{s}}^v + T_{\check{i},\check{s}}^e) \end{aligned}$$

Expressions (4a)-(4b) are equivalent to the desired objective $\min\max_{i\in\mathcal{R}} \hat{t}_i^x$, while equalities (4c) and (4d) define the values for $\hat{t}_i^x$ and $\hat{t}_i^e$, respectively. Assignment constraints (4e) and (4f) indicate that only one task is assigned to each robot and that each task is assigned to exactly one robot, respectively. Constraints (4g) - (4i) indicate whether robots i and k are in consecutive lanes which is expressed with the logical constraint, "if $|\sum_{s\in\mathcal{S}} x_{i,s}l_s - \sum_{s\in\mathcal{S}} x_{k,s}l_s| \leq 1$ then $\delta_{i,k} = 1$". This condition is equivalent to condition "if $(\sum_{s\in\mathcal{S}} x_{i,s}l_s - \sum_{s\in\mathcal{S}} x_{k,s}l_s \leq 1) \wedge (\sum_{s\in\mathcal{S}} x_{i,s}l_s - \sum_{s\in\mathcal{S}} x_{k,s}l_s \leq 1)$ then $\delta_{i,k} = 1$ which can be expressed using (4g) - (4i) according to LC4. Constraint (4j) ensures that if $\psi_{i,k} = 1$ then $\hat{t}_k^e \geq \hat{t}_i^e + T^g$ according to LC1. Constraint (4k) establishes the precedence between robots i and k; if the particular pair of robots is potentially conflicting ($\delta_{i,k} = 1$) then either $\hat{t}_k^e \geq \hat{t}_i^e + T^g$ or $\hat{t}_i^e \geq \hat{t}_k^e + T^g$ indicating that robot i precedes robot k with the necessary guard time and vice-versa. Constraints (4l) - (4n) ensure that if $\psi_{i,k} = 1$ then exactly one of the non-conflict conditions holds (LC2 and LC3); CFC case 1 holds when $\xi_{i,k} = 1$ and CFC case 2 when $\xi_{i,k} = 0$. Finally, constraints (4o) - (4q) ensure the non-negativity of the waiting times and the binary nature of the indicator and assignment variables.

The developed MILP solution approach guarantees optimality and serves as the baseline for the performance evaluation of other developed algorithms. In addition, it can be deployed as the preferred optimization approach in cases where the solution is available prior to the time that is needed. Nonetheless, the mixed-integer nature of the formulation implies that in certain cases the MILP solver may need exponentially large time to complete or reach a certain optimality gap. This is also empirical verified from the simulation results presented in Section VI, which indicate that the execution time of the MILP solver is quite large in certain cases. As the MILP optimization approach is not ideal for real-time optimization, we also develop a close-to-optimal low polynomial complexity heuristic in the next section.

V. LOW TIME-COMPLEXITY SOLUTION

In this section, we develop a low time-complexity heuristic approach suitable for very fast solution of the considered problem. The approach taken is to decouple the assignment and coordination problems and solve them sequentially.

To deal with the *assignment problem*, the objective is to find the best allocation of robots to tasks that minimizes the assignment cost, defined as

$$f_A = \max_{i\in\mathcal{R}} t_i^x,$$

which denotes the time at which all tasks have been completed ignoring potential conflicts; this issue will be addressed in the coordination problem. Therefore, w^e and w^x are always zero, and the time robot i takes to complete its task is now reduced to $t_i^x + T^d = t_i^e + 2t_i^v + T^l + T^d$. The cost matrix $\mathbf{C}$ has elements

$$c_{is} = T_{i,s}^e + 2T_s^v + T^l + T^d$$

which represent the time required by robot i to complete a candidate task s. The assignment problem is defined as

$$\begin{aligned}&\min_{\mathbf{X}} \max_{1\le i,s\le n} c_{is}x_{is}\\ \text{s.t.}\ &\sum_{i=1}^{n} x_{i,s}=1,\ s\in\mathcal{S},\\ &\sum_{s=1}^{n} x_{i,s}=1,\ i\in\mathcal{R},\\ &x_{i,s}\in\{0,1\},\ i\in\mathcal{R},\ s\in\mathcal{S}.\end{aligned}$$

This problem is equivalent to the *Linear Bottleneck Assignment Problem* (LBAP) [34], [35] which describes the assignment of n jobs to n machines such that the latest completion time is as early as possible. This problem can be solved using a threshold algorithm in $O(n^{2.5}/\sqrt{\log n})$ time [34, Theorem 6.4].

To deal with the *coordination problem*, it is important to find an efficient strategy that shifts and/or extends ATWs in order to complete all tasks with no conflicts, in order to minimize the coordination cost defined as

$$f_C=\max_{i\in\mathcal{R}}(w_i^e+w_i^x-\gamma_i),$$

where $\gamma_i=\max_{j\in\mathcal{R}} t_j^x-t_i^x$, $\gamma_i\ge 0$, is the waiting time that can be added without increasing the total cost. Next, we examine the two-robot problem in order to select an appropriate coordination strategy for the multi-robot problem. To solve the two-robot problem, four cases need to be considered depending on the relative arrangement of the corresponding ATWs: (i) $(t_i^e<t_k^e)\wedge(t_i^v>t_k^v)$ (ii) $(t_i^e>t_k^e)\wedge(t_i^v>t_k^v)$ (iii) $(t_i^e>t_k^e)\wedge(t_i^v<t_k^v)$ (iv) $(t_i^e<t_k^e)\wedge(t_i^v<t_k^v)$. Note that we only need to examine cases (i) and (ii), as cases (iii) and (iv) are their mirror cases which can be addressed by swapping indices i and k. Any initial configuration of two ATWs has 3 possible solutions with different performance (solution 1, 2 and 3) presented in Fig. 5. Solutions 1-3 are obtained by applying strategies 1-3, respectively:

Strategy 1 shifts the ATW with the smallest t^v and extends the ATW with the largest t^v

Strategy 2 only shifts the ATW with the smallest t^v

Strategy 3 only shifts the ATW with the largest t^v

Strategies will shift and/or extend the ATWs only when necessary, in order to achieve the best performance in terms of the objective value. The following lemmas are used to choose the best strategy for coordination.

Lemma 1: The best case scenario performance of Strategy 2 is better than the worst case scenario performance of Strategy 1 by $t_i^e-t_k^e+2T^g$ for case (i) and $T^g-2t_i^v-T^l$ for case (ii), given that the particular quantity is positive.

Proof: The proof can be found in Appendix B. ■

Lemma 1 indicates that Strategy 2 is better than Strategy 1 by at most $2T^g$ for case (i), in the best case; for case (ii) it is better only in the unlikely event that $T^g-2t_i^v-T^l>0$.

Lemma 2: The best case scenario performance of Strategy 3 is better than the worst case scenario performance of Strategy 1 by $t_i^e-t_k^e+T^g-2t_i^v-T^l$ for case (i), and $2t_k^v-2t_i^v+2T^g$ for case (ii).

Proof: The proof can be found in Appendix C. ■

Lemma 2 indicates that Strategy 3 is better than Strategy 1 for case (i) in the unlikely case where $T^g-2t_i^v-T^l>0$, as the conditions that govern the particular scenarios require $0<t_k^e-t_i^e<T^g$, and by at most $2T^g$ for case (ii).

Lemma 3: The best case scenario performance of Strategy 1 is better than the worst case scenario performance of Strategy 2 by $2t_k^v+T^l+T^g$ for case (i) and $2t_k^v+T^l+T^g$ for case (ii).

Proof: The proof can be found in Appendix D. ■

Lemma 4: The best case scenario performance of Strategy 1 is better than the worst case scenario performance of Strategy 3 by $t_k^e-t_i^e+2t_k^v+T^l+T^g$ for case (i) and $t_k^e-t_i^e+2t_k^v+T^l+T^g$ for case (ii).

Proof: The proof can be found in Appendix E. ■

Lemma 4 indicates that Strategy 1 is always better than Strategy 3 because $t_i^e<t_k^e$ for case (i) and $t_i^e<t_k^e+2t_k^v+T^l+T^g$ for case (ii)

Based on the above analysis, we have chosen Strategy 1 for the solution of the coordination problem. That is because, Lemmas 1 and 2 indicate that when Strategies 2 and 3 are better than Strategy 1, the additional time imposed if Strategy 1 is used instead, is bounded by $2T^g$ which is very small compared to the other parameters. Lemmas 3 and 4 further indicate that the benefit of using Strategy 1 instead of the other strategies is more substantial because it depends on the values of $t^v >> T^g$. In sum, Strategy 1 provides the best performance from the three strategies irrespective of the ATW configuration. Furthermore, by solely using Strategy 1 in the algorithm, the time-complexity is simplified when the problem scales up as $|\mathcal{R}|$ increases. Strategy 1 involves the sequential execution of the Shift and Extend operations; hence, our coordination algorithm also involves the sequential execution of these operations as described in Algorithm 1. Note that Algorithm 1 is not expected to always be optimal, as the optimal solution may involve a mixture of strategies regarding the relative arrangement of different ATWs.

In the ATW context, the first part of the algorithm resolves conflicts at the entering part of ATWs. The algorithm starts by sorting the robot ATWs with respect to the t^v time, in descending order i.e. the ATW with the larger t^v value is placed first. Starting from the first ATW in the set, the conflicting ATWs are determined and stored in the $\mathcal{C}_r$ set. Depending on which conditions hold, the appropriate waiting time w^e is calculated for robot ATW k. Sorting the ATWs in the $\mathcal{R}^{\overleftarrow{s}}$ reduces the complexity because when Shift is applied starting from the ATW with the largest t^v, ensures that a specific ATW will not have a conflict later and therefore needs to be checked only once. This applies because according to lemmas 3 and 4, when two ATWs are conflicting, it is more efficient to apply Shift (add waiting time w^e) to the ATW with the smallest t^v in order to resolve the conflict. Therefore, starting from the ATW with the largest t^v, any arising conflicts will affect ATWs that have yet to be examined; hence the first part of the algorithm will terminate in one iteration.

The second part of the algorithm resolves the conflicts on the exiting part of ATWs. This time, the t^v values are sorted in ascending order to obtain set $\mathcal{R}^{\overrightarrow{s}}$. The Extend operation is applied starting from the ATW with the smallest t^v, ensures

Algorithm 1 Coordination Algorithm

1: */* Operation Shift */*
2: $\mathcal{R}^{\overleftarrow{s}} = \text{sort}(\mathcal{R}, \mathbf{t}^v, \text{'descending'})$
3: **for** $r \in \mathcal{R}^{\overleftarrow{s}}$ **do**
4: **for** $c \in \mathcal{C}_r$ **do**
5: $i = \text{argmax}\{t_z^v | z \in \{r, c\}\}$;
6: $k = \text{argmin}\{t_z^v | z \in \{r, c\}\}$;
7: **if** $\hat{t}_i^e + T^g > \hat{t}_k^e$ **then**
8: $w_k^e = \hat{t}_i^e - \hat{t}_k^e + T^g$;
9: **else**
10: $w_k^e = 0$;
11: **end if**
12: **end for**
13: **end for**
14: */* Operation Extend */*
15: $\mathcal{R}^{\overrightarrow{s}} = \text{sort}(\mathcal{R}, \mathbf{t}^v, \text{'ascending'})$
16: **for** $r \in \mathcal{R}^{\overrightarrow{s}}$ **do**
17: **for** $c \in C_r$ **do**
18: $i = \text{argmax}_i\{t_z^v | z \in \{r, c\}\}$;
19: $k = \text{argmin}_i\{t_z^v | z \in \{r, c\}\}$;
20: **if** $\hat{t}_k^x + w_k^e + T^g > \hat{t}_i^x$ **then**
21: $w_i^x = \hat{t}_k^x + w_k^e - \hat{t}_i^x + T^g$;
22: **else**
23: $w_i^x = 0$;
24: **end if**
25: **end for**
26: **end for**

that an examined ATW need not be re-examined. This applies because according to lemmas 3 and 4, when two ATWs are conflicting, it is more efficient to apply Extend (add waiting time w^x) to the ATW with the largest t^v in order to resolve the conflict. Depending on what conditions hold between the two ATWs, waiting time w^x is calculated for robot ATW i. When this operation finishes, all ATWs are conflict free and the solution of the problem is reached.

The algorithm involves the Shift and Extend operations which are of the same time-complexity. Each operation requires sorting the elements which is of time-complexity $O(n \log n)$ and two nested for-loops each of time-complexity $O(n^2)$. Hence, total time-complexity of Algorithm 1 is dominated by the nested for-loops resulting in $O(n^2)$. Note that the original problem is solved using an LBAP assignment algorithm from the literature, and the novel coordination algorithm described in Algorithm 1.

To theoretically examine the performance of Algorithm 1 with respect to the optimal, we define f_A^*, f_C^* and $f_T^* = f_A^* + f_C^*$ as the assignment, coordination and total cost resulting from the optimal solution (obtained through MILP), respectively. Also, let f_A^h, f_C^h and $f_T^h = f_A^h + f_C^h$ denote the assignment, coordination and total cost resulting from LBAP and Algorithm 1, respectively. Next, we derive theoretical bounds for the assignment, coordination and total cost of the heuristic algorithm.

Lemma 5: For the assignment cost, it is true that $f_A^h \leq f_A^*$.

Proof: The heuristic algorithm solves the assignment problem using LBAP which by definition provides the minimum assignment cost. Hence, f_A^h is smaller than or equal to the assignment cost of any other algorithm that can be proposed to solve the problem including the MILP approach. ■

The fact that Lemma 5 ensures that the heuristic algorithm has minimum assignment cost, does not mean that it also has minimum total cost. Hence, although $f_A^h \leq f_A^*$ it is true that $f_T^* \leq f_T^h$.

Lemma 6: The solution of Algorithm 1 yields a coordination cost $f_C^h \leq 2(n-1)T^g$.

Proof: The proof can be found in Appendix F. ■

Theorem 1: The solution of the heuristic algorithm yields an additional total cost of at most $2(n-1)T^g$ when compared with the optimal i.e. $f_T^h \leq f_T^* + 2(n-1)T^g$.

Proof: From Lemma 5 it is true that $f_A^h \leq f_A^*$; also, from Lemma 6 it is true that $f_C^h \leq 2(n-1)T^g \leq 2(n-1)T^g + f_C^*$, as $f_C^* \geq 0$. Adding the two inequalities yields $f_A^h + f_C^h \leq f_A^* + f_C^* + 2(n-1)T^g$ which completes the proof. ■

Corollary 1: It is true that $f_T^h \to f_T^*$ as $T^g \to 0$.

The above results provide bounds for the performance of the heuristic algorithm with respect to the optimal and indicate that as the guard time tends to zero the performance tends towards optimality. The fact that a polynomial complexity algorithm yields the optimal solution for $T^g = 0$, indicates that the problem is not NP-hard for the particular case.

The reasoning behind Corollary 1 is that Strategy 1 yields $f_C^h = 0$ for any assignment in the limit where $T^g = 0$. This is achieved by shifting and extending certain ATWs resulting in an increase of certain t_i^x values; nonetheless, the increase is such that $t_i^x \leq t_{max}^x$, $i \in \mathcal{R}$, where $t_{max}^x = \max_{i \in \mathcal{R}}\{t_i^x\}$ is the maximum time to finish any task with no waiting. Hence, since the LBAP algorithm yields minimal assignment cost, the solution to the problem tends to optimality.

A. Case: $|\mathcal{L}| = 1$

A special case of the problem is when all tasks are located in a single lane. In this case the assignment problem is simplified and leads to faster computation [36]. To solve this problem efficiently we exploit the fact that every robot has to reach the same lane entrance. Therefore, we can derive the relative closeness of robots from the tasks based only on their distance from the entrance e.g. if i is closer to the entrance than k then i is closer to any task compared to k.

The assignment and coordination problems for the one-lane case are solved according to Algorithm 2. In this case the robot set is sorted with respect to the distance from the entrance such that $t_i^e < t_{i+1}^e$ defined as $\mathcal{R}^e$ (line 1 of Alg. 2). The task set is sorted as well such that $t_s^v > t_{s+1}^v$ defined as $\mathcal{S}^e$ (line 2). When these two vectors have this specific arrangement, the cost matrix $c_i = \sum_{s \in \mathcal{S}} T_{i,s}^e x_{i,s} + T^l + 2\sum_{s \in \mathcal{S}} T_s^v x_{i,s} + T^d$ fulfills the *bottleneck Monge property*. An n-by-m matrix A is called *bottleneck Monge matrix* if $\max\{a_{\gamma\delta}, a_{\theta\xi}\} \leq \max\{a_{\gamma\xi}, a_{\theta\delta}\}$ for all $1 \leq \gamma < \theta \leq n$, $1 \leq \delta < \xi \leq m$. Therefore the assignment matrix $\mathbf{X} = \mathbf{I}_n$ provides the optimal solution [34] by assigning the farthest task to the closest robot, the second farthest task to the second closest robot and so on (lines 3-5). The complexity of the assignment reduces to $O(n \log n)$ dominated by sorting.

Algorithm 2 Assignment and Coordination Algorithm for the One-lane Case

1: $\mathcal{R}^{\overrightarrow{e}} = \text{sort}(\mathcal{R}, \mathbf{t}^e, \text{'ascending'})$
2: $\mathcal{S}^e = \text{sort}(\mathcal{S}, \mathbf{t}^v, \text{'descending'})$
3: **for** $i = 1 \dots n$ **do**
4: $\quad x_{\mathcal{R}^{\overrightarrow{e}}(i), \mathcal{S}^e(i)} = 1$
5: **end for**
6: **for** $r \in \mathcal{R}^{\overrightarrow{e}}$ **do**
7: $\quad$ **if** $t^e_r + T^g > t^e_{r+1}$ **then**
8: $\quad\quad w^e_{r+1} = t^e_r - t^e_{r+1} + T^g$
9: $\quad$ **else**
10: $\quad\quad w^e_{r+1} = 0$
11: $\quad$ **end if**
12: **end for**
13: $\mathcal{R}^{\overleftarrow{e}} = \text{sort}(\mathcal{R}, \mathbf{t}^e, \text{'descending'})$
14: **for** $r \in \mathcal{R}^{\overleftarrow{e}}$ **do**
15: $\quad$ **if** $t^x_{r+1} + w^e_{r+1} + T^g > t^x_r$ **then**
16: $\quad\quad w^x_r = t^x_{r+1} + w^e_{r+1} - t^x_r + T^g$
17: $\quad$ **else**
18: $\quad\quad w^x_r = 0$
19: $\quad$ **end if**
20: **end for**

Note also that while Alg. 1 handles only the coordination problem, Alg. 2 handles both the assignment (lines 3-5) and coordination problems (lines 6-20). Although the coordination algorithm for this special case is similar to Alg. 1, operations 1 and 2 are simplified because all robots conflict with each other. This requires iteration through the robot list only once for each operation and hence the complexity of the coordination algorithm to $O(n)$. Hence, the overall complexity of Alg. 2 is $O(nlog(n))$.

B. Case: $|\mathcal{S}| \neq |\mathcal{R}|$

When the number of robots is not equal to the number of tasks ($|\mathcal{S}| \neq |\mathcal{R}|$) two cases can arise: (a) $|\mathcal{S}| < |\mathcal{R}|$, and (b) $|\mathcal{S}| > |\mathcal{R}|$. In the first case where the number of tasks is smaller than the number of robots, the heuristic algorithm remains almost identical. With respect to the assignment phase, the only difference is that $|\mathcal{R}| - |\mathcal{S}|$ "virtual" tasks need to be defined with cost equal to zero for all robot-task pairs, before the assignment algorithm can be applied. In the returned solution, the robots executing the virtual tasks will do nothing in the coordination phase as no task is assigned to them. The coordination phase will involve $|\mathcal{S}|$ robots and $|\mathcal{S}|$ tasks; hence, it can be dealt with using Alg. 1.

In the second case where the number of tasks is larger than the number of robots, the heuristic needs to be altered substantially as each robot needs to execute multiple tasks. In this case, challenges arise both in the assignment phase from the allocation of multiple tasks to each robot and in the coordination phase from the existence of loaded or unloaded robots in both the free-moving and storage areas complicating the conflict resolution significantly.

Alg. 3 outlines a rolling horizon strategy for solving this problem. The main idea behind this strategy is to solve

Algorithm 3 Assignment and Coordination when $|\mathcal{S}| > |\mathcal{R}|$

1: Compute $T^e_{i,s}$ based on initial locations of robots.
2: Set $c_{i,s} = T^e_{i,s} + 2T^v_s + T^l + T^d$;
3: Solve the assignment and coordination problems using LBAP and Alg. 1.
4: Set $t = 0$, $t^{st}_i = 0,\ i \in \mathcal{R}$ and $\mathcal{S}^{rem} = \mathcal{S}$.
5: **repeat**
6: $\quad$ Set $r = \text{argmin}_{i \in \mathcal{R}}\{t^{st}_i + \hat{t}^x_i\}$, $t = \min_{i \in \mathcal{R}}\{t^{st}_i + \hat{t}^x_i + T^d\}$.
7: $\quad$ Set $\mathcal{S}^{rem} \leftarrow \mathcal{S}^{rem} - \{S(r)\}$ where $S(r)$ denotes the task of robot r.
8: $\quad$ Define sets $\mathcal{R}^F$ and $\mathcal{R}^R$ at time t.
9: $\quad$ Determine times $t^c_i = t^{st}_i + \hat{t}^x_i - t$, $i \in \mathcal{R}^R$ to clear occupying lanes.
10: $\quad$ Set $t^a_l = \max_{i \in \mathcal{R}(l)}\{\max\{0,\ t^c_i + T^g\}\}$, $l \in \mathcal{L}$, where $\mathcal{R}(l) \subseteq \mathcal{R}^R$ is the set of robots in lane l or adjacent lanes.
11: $\quad$ Set $d_i = t^{st}_i + \hat{t}^x_i - t$, $i \in \mathcal{R}^R$ and $d_i = 0$, $i \in \mathcal{R}^F$.
12: $\quad$ Compute $c_{i,s} = \max\{t^a_{L(s)}, d_i + T^e_{i,s}\} + 2T^v_s + T^l + T^d$, where $L(s)$ is the lane of task s, and $T^e_{i,s}$ is computed based on the current locations of robots for $i \in \mathcal{R}^F$ or the locations of robots upon completing their current task for $i \in \mathcal{R}^R$.
13: $\quad$ Solve the assignment problem based on LBAP.
14: $\quad$ Solve the coordination problem for the new assignments using Alg. 1.
15: $\quad$ Set $t^{st}_i = t$ for $i \in \mathcal{R}^F$.
16: **until** $(|\mathcal{S}^{rem}| = 0)$

the assignment and coordination problem every time a robot completes a task, taking into consideration all robots even if they are currently executing a task. For the set of robots that are unloaded and in the free-moving area, $\mathcal{R}^F$, the newly assigned task cancels the current assignment, while for the set of robots that are *reserved*, $\mathcal{R}^R$, (i.e. loaded or located in the storage area) the newly assigned task has to be executed upon completion of the current assignment. Moreover, the additional time from time t before entrance in lane l is allowed, t^a_l, is computed which depends on the time needed for reserved robots in lane l and adjacent lanes to clear the storage area, t^c_i, $i \in \mathcal{R}^R$; note that $t^c_i < 0$ for robots that are loaded and in the free-moving area. For each reserved robot $i \in \mathcal{R}^R$, time d_i also needs to be computed which accounts for the time needed to complete their current task in order to start a new task. Note that the entire procedure outlined in Alg. 3 is executed prior to the start of the movement of the robots. In addition, the rolling horizon approach is a potential solution for the case $|\mathcal{S}| > |\mathcal{R}|$, but requires further study to demonstrate its efficiency in a range of settings, which is beyond the scope of this paper.

VI. Simulation Results

In this section we present results for evaluating the performance of the proposed method. Simulations were executed on a Intel Core i7-4790K CPU at 4.0GHz with 16GB of RAM. For the modeling and solution of the MILP formulation the Gurobi Optimizer 6.5.2 mathematical programming solver was used [37], while the parameters M^l_2 and M^u_1 are

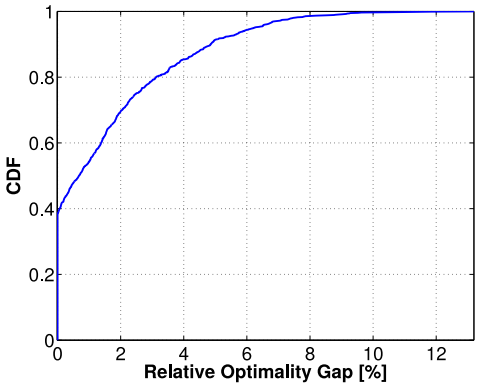


Fig. 7. The cumulative distribution function of the relative optimality gap between the optimal and heuristic solution.

defined according to the analysis of Section IV. The simulated warehouse was designed based on the topology of Fig. 2. The length of the warehouse is 300 m, with the lower 100 m being the free- moving space. The width of each lane is 3 m and they are spaced 4 m apart leaving 1 m for the aisles. The warehouse width depends on the number of lanes used and because that number varies in our experiments so does the warehouse width. To compare the performance of the heuristic algorithm relative to the MILP one we employ the relative optimality gap metric which is defined as:

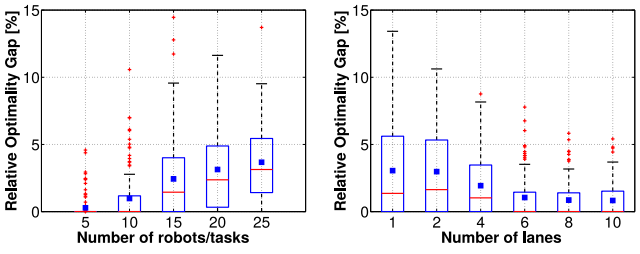
$$\text{Relative Optimality Gap} = \left(\frac{f_T^h}{f_T^*} - 1\right) \times 100\%$$

Figure 7 is the cumulative distribution function of the relative optimality gap between the optimal and heuristic solution, which is the result of a set of 1200 problems with $n = 20$ and 10 lanes. The heuristics algorithm solved 40% of the problems optimally and 95% of them with at most 6% optimality gap.

Figure 8(a) demonstrates the relative optimality gap with respect to the number of robots in the form of a box-plot[2]. To obtain the results, we simulated 200 problems for each $n = \{5, 10, 15, 20, 25\}$. The initial conditions of each simulation i.e. the positions of robots and tasks, were chosen randomly. The number of lanes is 4 and is kept constant for all simulations. In all considered cases the mean optimality gap is less than 4%, while 75% and 100% of the problems in each case have relative optimality gap within 6% and 15%, respectively. As the number of robots increases the relative optimality gap also increases. This is because as the number of robot increases, more robots have to move in conflicting lanes resulting in more conflicts, making the problem harder to solve.

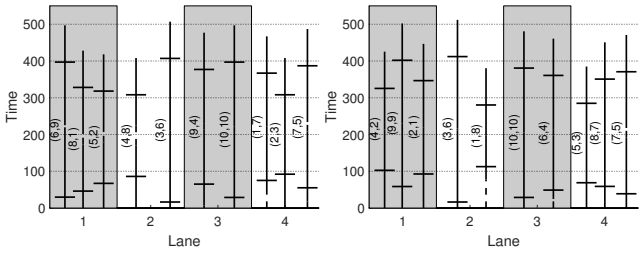
The next experiment demonstrates the relative optimality gap with respect to the number of lanes while the number of robots is kept constant at $n = 15$. The results presented in Fig. 8(b), show that 75% of all problems in each case have at

[2]The bottom and top of each box indicate the first and third quartiles (25% and 75%) of a ranked data set, while the horizontal line inside the box indicates the median value (second quartile). The horizontal lines outside the box indicate the lowest/highest datum still within 1.5 inter-quartile range of the lower/upper quartile; for normally distributed data this corresponds to approximately 0.35%/99.65%.



(a) (b)

Fig. 8. Relative optimality gap with respect to the number of (a) robots/tasks, and (b) lanes.



(a) (b)

Fig. 9. Solution comparison of the same scenario between (a) MILP solver and (b) the heuristic. Numbers in parenthesis correspond to the robot ID and task ID respectively.

most 6% gap, while the mean value of each group is below 3%. As the number of lanes decreases i.e. the average number of robots per lane increases (similarly to Fig 8(a)), it causes more conflicts and therefore more gap between the optimal and heuristic solution.

Simulation results shown in Fig. 9 demonstrates the assignment and coordination solution of one problem instance when $|\mathcal{S}| = |\mathcal{R}| = 10$ and $|\mathcal{L}| = 4$, using (a) the MILP solver, and (b) the heuristic algorithm. In the chosen scenario, the optimal objective value, obtained through MILP, is equal to 507, while the heuristic yields a cost of 512. The slightly better solution produced by the MILP solver was achieved by first assigning robot 9 rather than 6 (of the heuristic) to task 4. In comparison to the heuristic solution, this assignment causes a slight delay in delivering task 4, but improves the delivery time of task 9 which is now assigned to robot 6. This ultimately improves the objective value since robot 3 delivering task 6 benefits from the improved delivery time of task 9.

Figure 10 depicts relative optimality gap for varying guard time. It can be seen that the relative optimality gap drops from 5% at $T^g = 32$ to 0.004% at $T^g = 0.1$, about three orders of magnitude. This behavior is in agreement with the theoretical result that as the guard time decreases the performance of our heuristic algorithm tends to optimality (Corollary 1).

Execution time is an important criterion for the ability of an algorithm to perform in real time. The results presented in Fig. 11(a), show that in about 40% of problems the MILP solver required more that 50 s to reach a solution. Note that the reported MILP execution time is purely solver time and does not include the mathematical modeling time. Also about 8% of problems took between 200 and 1800 s while about 15% of problems did not finish within the time limit of 30 minutes. The results are significantly better when a 5% optimality gap is allowed, with 95% of the problems solved within 250s.

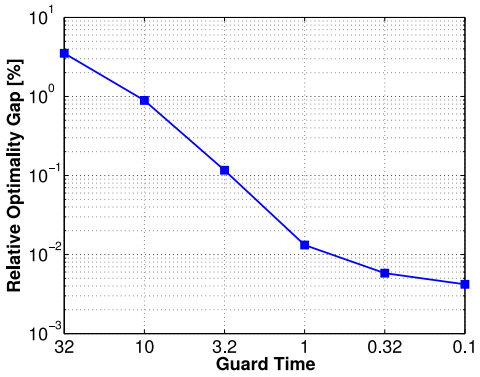
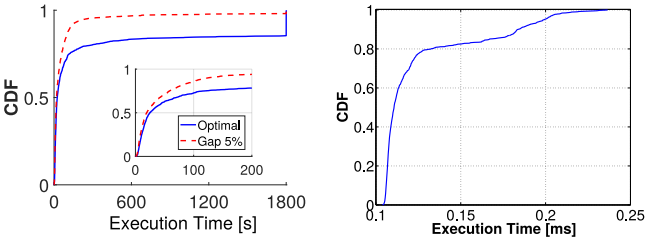


Fig. 10. Relative optimality gap for varying guard time; the results are averaged over 200 random problems with $n = 15$ and 10 lanes.

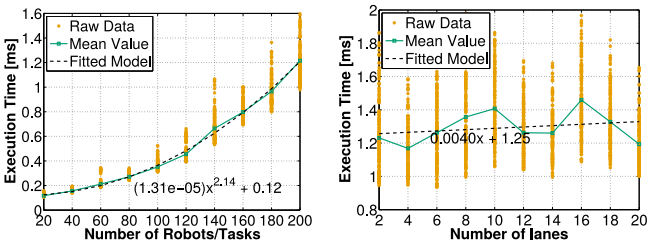


(a) (b)

Fig. 11. Cumulative distribution function of the execution time of the (a) MILP solver, and (b) proposed heuristic solution.

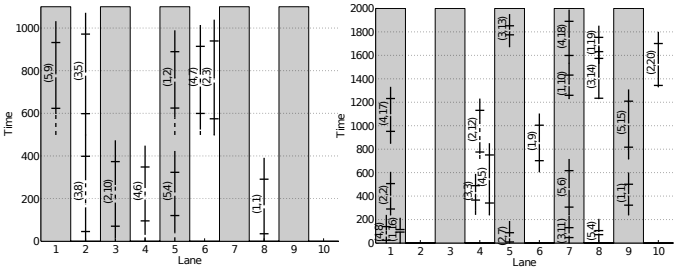
However, the rest 5% require a significantly larger execution time, while about 1% of the problems are not solved within the 30 minute deadline. These results indicate that the MILP solver is not ideal for real-time applications due to the large execution times and unpredictable performance, evident from the large variability of the execution time of all problems. Figure 11(b) shows the heuristic performance on the same problem set of Fig. 11(a). All problems were solved by the algorithm within 0.25 ms, with fastest solution reached within 0.1 ms. Hence, the developed algorithm is very fast and with small execution time variability making it suitable for real-time applications. Compared to the MILP solver, the proposed algorithm is over one million times faster (six-orders of magnitude).

To demonstrate that the implemented algorithm complexity follows the theoretical complexity analysis, execution times were recorded for 200 simulations of varying n as shown in Fig 12(a). Each point in the figure denotes the execution time for each simulation, while the solid line depicts the mean values which illustrates the increase trend. The dashed line shows the nonlinear least-squares fit of the polynomial model $a_1 x^{a_2} + a_3$. The fitted value of the exponent $a_2 = 2.14$ indicates that the execution time scales almost quadratically to the number of robots/tasks, as expected from the theoretical time-complexity analysis in Section V. Figure 12(b) is the result of the same method but with varying lanes number. As shown from the characteristics of the fitted model, the execution time is almost independent from the number of lanes. Finally, Figs. 12(a) and 12(b) show that the developed algorithm can solve problems with a large number of robots/tasks or lanes very fast (in less than 1.5 ms for 200 robots), illustrating scalability and potential for real-time deployment.



(a) (b)

Fig. 12. Heuristic algorithm execution time with respect to the number of (a) robots/tasks, and (b) lanes. Each point on the graph represents a different simulation.



(a) (b)

Fig. 13. Solution illustration with ATWs, for the case $|\mathcal{S}| > |\mathcal{R}|$ using Alg. 3. In both problems $|\mathcal{R}| = 5$, $T^l = 20$, $T^g = 25$. (a) $|\mathcal{S}| = 10$ and (b) $|\mathcal{S}| = 20$. Note that all ATWs satisfy the CFC i.e. there are no conflicts between the robots.

The simulation results shown in Fig. 13 demonstrate the solution to the problem when the number of task is larger than the number of robots i.e. $|\mathcal{S}| > |\mathcal{R}|$. As shown in the figures, after a robot delivers its task to the loading station, it proceeds to pick-up the next one. A robot does not have to wait for the rest of the robots to deliver their tasks; Alg. 3 will immediately assign a new task to any robot that delivers its task. The algorithm considers the current task and position of all robots in order to make an efficient assignment and coordination decision. Figure 13(a) shows that robots deliver their tasks in "waves" of five (which is the number of robots in the simulation) i.e. there is a distinct separation between the delivery of the first five tasks from the following five tasks. This happens because the spatial separation of the robots is small in the early stages of the simulation. However, when more time is allowed, the robots break-out of the "waves" pattern and work independently. In Fig. 13(b) for example, where the number of tasks 20, after a couple of iterations robot 3 happens to be in a more suitable position therefore by the end of the simulation delivers more tasks than the rest of the robots.

VII. Conclusions

In this work we investigate a problem associated with transferring a set of containers from the storage to the loading area of a warehouse using autonomous robots. We define the considered problem incorporating the constraints imposed by the topology. The problem is formulated and solved optimally using Mixed Integer Linear Programming tools. Furthermore, a low time-complexity heuristic algorithm is developed and theoretically investigated. It is shown that its performance relative to optimality is upper bounded by the product of the number of robots and the guard time, implying that the heuristic tends

to optimality when the guard time tends to zero. Simulation results indicate that the developed heuristic approach provides solutions on average within 5% relative optimality gap, with about 40% of the problems solved optimally. In terms of execution time, the heuristic approach executes in the order of milliseconds and is six orders of magnitude faster than a state-of-the-art Mixed Integer Linear Programming solver.

An interesting future direction is to determine the complexity class of the problem for different cases. This work has shown that the developed polynomial complexity algorithm solves the problem optimally when there is no guard time; nonetheless, the complexity of the problem in the general case is still an open question.

APPENDIX

A. Analysis of Strategies

This section analyses the performance of the three strategies in all possible conditions. This analysis is used for the derivation of Lemmas 1 – 4; hence, it is presented in advance.

Strategy 1 shifts the ATW with the smallest t_k^v and extends the ATW with the largest t_i^v to achieve the solution illustrated in Fig. 5(a). First, the condition C1: $t_k^e - t_i^e < T^g$ is used to determine whether waiting time w_k^e is needed. A second condition is then needed to determine whether waiting time w_i^x is needed. This condition is either C2: $t_i^x < t_k^x + w_k^e + T^g$ or C3: $t_i^x > t_k^x + T^g$ depending on C1 being True or False, respectively. This leads to four possible combinations: (a) C1&C2, yielding $\mathbf{w}^e = [0, t_i^e - t_k^e + T^g]^\top$, $\mathbf{w}^x = [2t_k^v - 2t_i^v + 2T^g, 0]^\top$ and $\Lambda_{1a} = t_i^x + 2t_k^v - 2t_i^v + 2T^g$ (b) C1&$\overline{\text{C2}}$ yielding $\mathbf{w}^e = [0, t_i^e - t_k^e + T^g]^\top$, $\mathbf{w}^x = [1, 0]^\top$ and $\Lambda_{1b} = t_i^x$ (c) $\overline{\text{C1}}$&C3 yielding $\mathbf{w}^e = [0, 0]^\top$, $\mathbf{w}^x = [t_k^x - t_i^x + T^g, 0]^\top$ and $\Lambda_{1c} = t_k^x + T^g$ (d) $\overline{\text{C1}}$&$\overline{\text{C3}}$ yielding $\mathbf{w}^e = [0, 0]^\top$, $\mathbf{w}^x = [0, 0]^\top$ and $\Lambda_{1d} = t_i^x$.

Strategy 2 shifts only the ATW with the smallest t^v to achieve the solution illustrated in Fig. 5(b). Condition C4 : $t_i^x + T^g > t_k^e$ is used to determine whether w_k^e is needed. This leads to two possible outcomes: (a) C4, yielding $\mathbf{w}^e = [0, t_i^x - t_k^e + T^g]^\top$, $\mathbf{w}^x = [0, 0]^\top$ and $\Lambda_{2a} = t_k^x + t_i^x - t_k^e + T^g$ (b) $\overline{\text{C4}}$, yielding $\mathbf{w}^e = [0, 0]^\top$, $\mathbf{w}^x = [0, 0]^\top$ and $\Lambda_{2b} = t_k^x$.

Strategy 3 shifts only the ATW with the largest t^v to achieve the solution illustrated in Fig. 5(c). The solution of strategy 3 is non-conditional yielding $\mathbf{w}^e = [t_k^x - t_i^e + T^g]^\top$, $\mathbf{w}^x = [0, 0]^\top$ and $\Lambda_3 = t_i^x + t_k^x - t_i^e + T^g$.

For case (ii), **Strategy 1** always requires waiting time $\mathbf{w}^e = [0, t_i^e - t_k^e + T^g]^\top$ and using the condition C5: $t_k^x + w_k^e + T^g > t_e^x$ it determines whether exit waiting time is needed. This leads to two possible outcomes (a) C1, yielding $\mathbf{w}^x = [2t_k^v - 2t^v i + 2T^g, 0]^\top$ and $\Lambda_{1a} = t_i^x + 2t_k^x - 2t_i^v + 2T^g$ (b) $\overline{\text{C1}}$, yielding $\mathbf{w}^x = [0, 0^\top]$ and $\Lambda_{1b} = t_i^x$.

The solution of **Strategy 2** is non-conditional yielding $\mathbf{w}^e = [0, t_i^x - t_k^e + T^g]^\top$, $\mathbf{w}^x = [0, 0]^\top$ and $\Lambda_2 = t_i^e + 2t_i^v + 2t_k^v + 2T^l + T^g$.

Finally for **Strategy 3** condition C6 is required to determine whether waiting time w_i^e is needed. This leads to two possible outcomes (a) C6, yielding $\mathbf{w}^e = [t_k^x - t_i^e + T^g, 0]^\top$, $\mathbf{w}^x = [0, 0]^\top$ and $\Lambda_{3a} = t_k^e + 2t_k^v + t_i^v + 2T^l + T^g$ (b) $\overline{\text{C6}}$, yielding $\mathbf{w}^e = [0, 0]^\top$, $\mathbf{w}^x = [0, 0]^\top$ and $\Lambda_{3b} = t_i^x$.

B. Proof of Lemma 1

For the initial conditions of case (i), the best case scenario for Strategy 2 is when $\overline{\text{C4}}$ occurs, and the worst case scenario for Strategy 1 is when C1&C2 occurs. Comparing the solutions of the two scenarios, yields $\Lambda_{1a} - \Lambda_{2b} = t_i^e - t_k^e + 2T^g$. For case (ii), the solution of Strategy 2 is non-conditional and the worst case scenario for Strategy 1 is when C5 occurs. Hence, we have that $\Lambda_{1a} - \Lambda_2 = T^g - 2t_i^v - T^l$.

C. Proof of Lemma 2

For the initial conditions of case (i), Strategy 3 solution is non-conditional therefore it always yields Λ_3. The worst case scenario for Strategy 1 is when C1&C2 occurs. Comparing the two, provides $\Lambda_{1a} - \Lambda_3 = -2t_i^v - T^l + T^g - (t_k^e - t_i^e)$, where $0 < T^g - (t_k^e - t_i^e) < T^g$. For case (ii), the best case scenario for Strategy 3 is when $\overline{\text{C6}}$ occurs. For the same conditions, the worst case scenario for Strategy 1 is when C5 occurs. Comparing the two solutions, Strategy 3 performs better by $\Lambda_{1a} - \Lambda_3 = 2t_k^v - 2t_i^v + 2T^g$.

D. Proof of Lemma 3

For case (i), the best case scenario for Strategy 1 is when C1&$\overline{\text{C2}}$ occurs. For the same conditions, worst case scenario for Strategy 2 is when C4 occurs. Strategy 1 performs better by $\Lambda_2 - \Lambda_1 = 2t_k^v + T^l + T^g$. For case (ii), best case scenario for Strategy 1 is when $\overline{\text{C5}}$ occurs and the solution for Strategy 2 is non-conditional. Therefore, Strategy 1 solution is better by $\Lambda_2 - \Lambda_1 = 2t_k^v + T^l + T^g$.

E. Proof of Lemma 4

For case (i), the best case scenario for Strategy 1 is when C1&$\overline{\text{C2}}$ occurs, and the solution for Strategy 3 is non-conditional. Therefore, Strategy 1 performs better by $\Lambda_3 - \Lambda_1 = t_k^e - t_i^e + 2t_k^v + T^l + T^g$ which is always positive because of the case (ii) condition $t_i^e < t_k^e$. For case (ii), best case scenario solution for Strategy 1 is is when $\overline{\text{C5}}$ occurs. For the same conditions, worst case scenario for Strategy 3 is when $\overline{\text{C6}}$ occurs. Strategy 1 performs better by $\Lambda_3 - \Lambda_1 = t_k^e - t_i^e + 2t_k^v + T^l + T^g$.

F. Proof of Lemma 6

Let us define $\hat{t}_{i,m}^e = t_i^e + \sum_{\tau=1}^m w_{i,\tau}^e$ and $\hat{t}_{i,m}^x = \hat{t}_{i,m}^e + 2t_i^v + \sum_{\tau=1}^m w_{i,\tau}^x + T^l$, where $w_{i,\tau}^e$ and $w_{i,\tau}^x$ are the entrance and exit waiting time of ATW i in step τ of the algorithm, respectively. Note that the m-th external for-loop iteration of the Shift and Extend operations are the m-th and $(n+m)$-th step of the algorithm, respectively.

Algorithm 1 is comprised of the Shift and Extend operations. The Shift operation examines ATWs in descending t_i^v value order; assuming that ATW i is examined during the m-th iteration of the Shift operation, other ATWs belonging to the set $j \in \mathcal{C}_i$ are appropriately shifted ($w_{j,m}^e > 0$ and $w_{j,m}^x = 0$) to ensure that $\hat{t}_{i,m}^e \leq \hat{t}_{j,m}^e - T^g$, $j \in \mathcal{C}_i$ and $t_i^v > t_j^v$. This condition is ensured by appropriately shifting ATW j by

$$w_{j,m}^e = \max(0, \hat{t}_{i,m-1}^e - \hat{t}_{j,m-1}^e + T^g). \tag{5}$$

To show that the total cost increase for this ATW pair is bounded by T^g, it suffices to prove that

$$\hat{t}^x_{j,m} - \max(\hat{t}^x_{i,m-1}, \hat{t}^x_{j,m-1}) \leq T^g. \quad (6)$$

In the case that $\hat{t}^x_{i,m-1} > \hat{t}^x_{j,m-1}$, condition (6) can be written as $\hat{t}^e_{j,m} + 2t^v_j - \hat{t}^e_{i,m-1} - 2t^v_i \leq T^g$. After substituting $\hat{t}^e_{j,m} = \hat{t}^e_{j,m-1} + w^e_{j,m}$ the equation is $\hat{t}^e_{j,m-1} + 2t^v_j + \max(0, \hat{t}^e_{i,m-1} - \hat{t}^e_{j,m-1} + T^g) - \hat{t}^e_{i,m-1} - 2t^v_i \leq T^g$ which is always true because $t^v_j - t^v_i < 0$. In the case that $\hat{t}^x_{i,m-1} < \hat{t}^x_{j,m-1}$, condition (6) can be written as $\hat{t}^x_{j,m} - \hat{t}^x_{j,m-1} = w^e_{j,m} \leq T^g$. By substituting $\hat{t}^e_{\kappa,m-1} = \hat{t}^x_{\kappa,m-1} - 2t^v_\kappa$, $\kappa = \{i, j\}$, into (5) yields $\max\left(0, \hat{t}^x_{i,m-1} - \hat{t}^x_{j,m-1} + 2(t^v_j - t^v_i) + T^g\right) \leq T^g$, which is true because $\hat{t}^x_{i,m-1} - \hat{t}^x_{j,m-1} < 0$ and $2(t^v_j - t^v_i) < 0$.

The Extend operation examines ATWs in ascending t^v_i value order; assuming that ATW i is examined during the m-th iteration of the Extend operation, i.e. step $l = n + m$ of the algorithm, other ATWs belonging to the set $j \in \mathcal{C}_i$ are appropriately extended ($w^x_{j,l} > 0$ and $w^e_{j,l} = 0$) to ensure that $\hat{t}^x_{i,l} \geq \hat{t}^x_{j,l} + T^g$, $j \in \mathcal{C}_i$ and $t^v_i > t^v_j$. This condition is ensured by appropriately shifting ATW j by

$$w^x_{i,l} = \max(0, \hat{t}^x_{i,l-1} - \hat{t}^x_{j,l-1} + T^g). \quad (7)$$

To show that the total cost increase for this ATW pair is bounded by T^g, it suffices to prove that

$$\hat{t}^x_{i,l} - \max(\hat{t}^x_{i,l-1}, \hat{t}^x_{j,l-1}) \leq T^g. \quad (8)$$

In the case that $\hat{t}^x_{i,l-1} > \hat{t}^x_{j,l-1}$, condition (8) can be written as $\hat{t}^x_{i,l} - \hat{t}^x_{i,l-1} = w^x_{i,l} \leq T^g$; this is equivalent to $\max(0, \hat{t}^x_{j,l-1} - \hat{t}^x_{i,l-1} + T^g) \leq T^g$ which is true because $\hat{t}^x_{j,l-1} - \hat{t}^x_{i,l-1} < 0$. In the case that $\hat{t}^x_{i,l-1} < \hat{t}^x_{j,l-1}$, condition (8) can be written as $\hat{t}^x_{i,l-1} - \hat{t}^x_{j,l} + \max(0, \hat{t}^x_{j,l-1} - \hat{t}^x_{i,l-1} + T^g) \leq T^g$ which is true because $\hat{t}^x_{i,l-1} < \hat{t}^x_{j,l} = \hat{t}^x_{j,l-1}$.

The algorithm needs $2n$ steps in order to complete and results in a maximum cost increase of $2(n-1)T^g$ as in each phase only $n-1$ pairwise comparisons are needed. Hence, the result $f^h_C \leq 2(n-1)T^g$.

REFERENCES

[1] H. Fazlollahtabar and M. Saidi-Mehrabad, "Methodologies to optimize automated guided vehicle scheduling and routing problems: a review study," *Journal of Intelligent & Robotic Systems*, vol. 77, no. 3-4, pp. 525–545, 2015.

[2] H. J. Carlo, I. F. Vis, and K. J. Roodbergen, "Transport operations in container terminals: Literature overview, trends, research directions and classification scheme," *European Journal of Operational Research*, vol. 236, no. 1, pp. 1–13, 2014.

[3] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, p. 9, 2008.

[4] L. Sabattini, V. Digani, C. Secchi, G. Cotena, D. Ronzoni, M. Foppoli, and F. Oleari, "Technological roadmap to boost the introduction of AGVs in industrial applications," in *Proceedings of IEEE International Conference on Intelligent Computer Communication and Processing*, 2013.

[5] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[6] G. Laporte, "Fifty years of vehicle routing," *Transportation Science*, vol. 43, no. 4, pp. 408–416, 2009.

[7] L. Qiu, W.-J. Hsu, S.-Y. Huang, and H. Wang, "Scheduling and routing algorithms for AGVs: a survey," *International Journal of Production Research*, vol. 40, no. 3, pp. 745–760, 2002.

[8] P. J. Egbelu and J. M. Tanchoco, "Potentials for bi-directional guide-path for automated guided vehicle based systems," *International Journal of Production Research*, vol. 24, no. 5, pp. 1075–1097, 1986.

[9] R. Gaskins and J. M. Tanchoco, "Flow path design for automated guided vehicle systems," *International Journal of Production Research*, vol. 25, no. 5, pp. 667–676, 1987.

[10] Y. A. Bozer and M. M. Srinivasan, "Tandem configurations for automated guided vehicle systems and the analysis of single vehicle loops," *IIE Transactions*, vol. 23, no. 1, pp. 72–82, 1991.

[11] K. S. Kim and B. Do Chung, "Design for a tandem AGV system with two-load AGVs," *Computers & Industrial Engineering*, vol. 53, no. 2, pp. 247–251, 2007.

[12] R. Z. Farahani, G. Laporte, E. Miandoabchi, and S. Bina, "Designing efficient methods for the tandem AGV network design problem using tabu search and genetic algorithm," *International Journal of Advanced Manufacturing Technology*, vol. 36, no. 9-10, pp. 996–1009, 2008.

[13] S. A. Reveliotis, "Conflict resolution in AGV systems," *IIE Transactions*, vol. 32, no. 7, pp. 647–659, 2000.

[14] R. L. Moorthy, W. Hock-Guan, N. Wing-Cheong, and T. Chung-Piaw, "Cyclic deadlock prediction and avoidance for zone-controlled AGV system," *International Journal of Production Economics*, vol. 83, no. 3, pp. 309–324, 2003.

[15] N. Wu and M. Zhou, "Deadlock resolution in automated manufacturing systems with robots," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 3, pp. 474–480, 2007.

[16] T. Brunsch, J. Raisch, and L. Hardouin, "Modeling and control of high-throughput screening systems," *Control engineering practice*, vol. 20, no. 1, pp. 14–23, 2012.

[17] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Ensemble coordination approach in multi-agv systems applied to industrial warehouses," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 922–934, 2015.

[18] S. C. Daniels, "Real time conflict resolution in automated guided vehicle scheduling," Ph.D. dissertation, Dept. of Industrial Eng., Penn. State University, PA, USA, 1988.

[19] C. W. Kim and J. M. Tanchoco, "Conflict-free shortest-time bidirectional AGV routeing," *International Journal of Production Research*, vol. 29, no. 12, pp. 2377–2391, 1991.

[20] J. Huang, U. Palekar, and S. Kapoor, "A labeling algorithm for the navigation of automated guided vehicles," *Journal of Engineering for Industry;(United States)*, vol. 115, no. 3, 1993.

[21] R. H. Möhring, E. Köhler, E. Gawrilow, and B. Stenzel, "Conflict-free real-time AGV routing," in *Proceedings of Operations Research*, 2005, pp. 18–24.

[22] N. Smolic-Rocak, S. Bogdan, Z. Kovacic, and T. Petrovic, "Time windows based dynamic routing in multi-AGV systems," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 151–155, 2010.

[23] C. W. Kim and J. M. Tanchoco, "Operational control of a bidirectional automated guided vehicle system," *International Journal of Production Research*, vol. 31, no. 9, pp. 2123–2138, 1993.

[24] C.-C. Lee and J. T. Lin, "Deadlock prediction and avoidance based on petri nets for zone-control automated guided vehicle systems," *International Journal of Production Research*, vol. 33, no. 12, pp. 3249–3265, 1995.

[25] C.-O. Kim and S. Kim, "An efficient real-time deadlock-free control algorithm for automated manufacturing systems," *International Journal of Production Research*, vol. 35, no. 6, pp. 1545–1560, 1997.

[26] N. Wu and M. Zhou, "Shortest routing of bidirectional automated guided vehicles avoiding deadlock and blocking," *IEEE/ASME Transactions on Mechatronics*, vol. 12, no. 1, pp. 63–72, 2007.

[27] D. Steenken, S. Voß, and R. Stahlbock, "Container terminal operation and operations research-a classification and literature review," *OR Spectrum*, vol. 26, no. 1, pp. 3–49, 2004.

[28] H. Martínez-Barberá and D. Herrero-Pérez, "Autonomous navigation of an automated guided vehicle in industrial environments," *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 4, pp. 296–311, 2010.

[29] H. Durrant-Whyte, D. Pagac, B. Rogers, M. Stevens, and G. Nelmes, "Field and service applications-an autonomous straddle carrier for movement of shipping containers-from research to operational autonomous systems," *IEEE Robotics & Automation Magazine*, vol. 14, no. 3, pp. 14–23, 2007.

[30] S. A. Reveliotis and E. Roszkowska, "Conflict resolution in free-ranging multivehicle systems: A resource allocation paradigm," *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 283–296, 2011.

[31] I. F. Vis, "Survey of research in the design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 170, no. 3, pp. 677–709, 2006.

[32] S. E. Ramaswamy and S. B. Joshi, "Deadlock-free schedules for automated manufacturing workstations," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 3, pp. 391–400, 1996.
[33] L. Magatao, "Mixed integer linear programming and constraint logic programming: towards a unified modeling framework," Ph.D. dissertation, University of Curitiba, May 2005.
[34] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2009.
[35] D. R. Fulkerson, I. Glicksberg, and O. Gross, *A production line assignment problem*. Sta. Monica, CA: The Rand Corporation, 1953.
[36] D. Stavrou and C. Panayiotou, "Task assignment and agent coordination in a warehouse environment," in *Proceedings of 20th Mediterranean Conference on Control & Automation*, 2012, pp. 1341–1346.
[37] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2015. [Online]. Available: http://www.gurobi.com

Demetris Stavrou has received M.Eng. in Electrical and Electronic Engineering in 2005 and M.Sc. in Modern Digital Communication Systems in 2006 from the University of Sussex, UK. He has received his Ph.D. degree in Electrical and Computer Engineering from the University of Cyprus in 2016. He is currently a senior researcher at Phoebe Innovations, Nicosia, Cyprus. He is a member of IEEE and IEEE Robotics and Automation Society.

Stelios Timotheou (S'04-M'10-SM'17) received a B.Sc. from the Electrical and Computer Engineering (ECE) School of the National Technical University of Athens, and an M.Sc. and Ph.D. from the Electrical and Electronic Engineering Department of Imperial College London. He is currently a Research Associate at the KIOS Research Center for Intelligent Systems and Networks of the University of Cyprus (UCY). In previous appointments, he was a Visiting Lecturer at the ECE Department of UCY, a Research Associate at the Computer Laboratory of the University of Cambridge and a Visiting Scholar at the Intelligent Transportation Systems Center & Testbed, University of Toronto. His research focuses on the modeling and system-wide solution of problems in complex and uncertain environments that require real-time and close to optimal decisions by developing optimization, machine learning and computational intelligence techniques.

Christos G. Panayiotou is an Associate Professor with the Electrical and Computer Engineering (ECE) Department at the University of Cyprus (UCY). He is also the Deputy Director of the KIOS Research Center for Intelligent Systems and Networks for which he is also a founding member. Christos has received a B.Sc. and a Ph.D. degree in Electrical and Computer Engineering from the University of Massachusetts at Amherst, in 1994 and 1999 respectively. He also received an MBA from the Isenberg School of Management, at the aforementioned university in 1999. Before joining UCY in 2002, he was a Research Associate at the Center for Information and System Engineering (CISE) and the Manufacturing Engineering Department at Boston University (1999 - 2002). His research interests include distributed and intelligent control systems, wireless, ad hoc and sensor networks, computer communication networks, fault diagnosis, optimization and control of discrete-event systems, resource allocation, transportation networks and intelligent buildings. Christos has published more than 190 papers in international refereed journals and conferences and is the recipient of the 2014 Best Paper Award for the journal Building and Environment (Elsevier). He is an Associate Editor for the Conference Editorial Board of the IEEE Control Systems Society, the IEEE Transactions on Control Systems Technology, the Journal of Discrete Event Dynamical Systems and the European Journal of Control. He held several positions in organizing committees and technical program committees of numerous international conferences.

Marios M. Polycarpou is a Professor of Electrical and Computer Engineering and the Director of the KIOS Research Center for Intelligent Systems and Networks at the University of Cyprus. He received the B.A degree in Computer Science and the B.Sc. in Electrical Engineering, both from Rice University, USA in 1987, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Southern California, in 1989 and 1992 respectively. His teaching and research interests are in intelligent systems and networks, adaptive and cooperative control systems, computational intelligence, fault diagnosis and distributed agents. Dr. Polycarpou has published more than 300 articles in refereed journals, edited books and refereed conference proceedings, and co-authored 7 books. He is also the holder of 6 patents. Prof. Polycarpou is a Fellow of IEEE and IFAC. He is the recipient of the 2016 IEEE Neural Networks Pioneer Award and the 2014 Best Paper Award for the journal Building and Environment (Elsevier). He has served as the President of the IEEE Computational Intelligence Society (2012-2013), and as the Editor-in-Chief of the IEEE Transactions on Neural Networks and Learning Systems (2004-2010). He is currently the Vice President of the European Control Association (EUCA). Prof. Polycarpou has participated in more than 60 research projects/grants, funded by several agencies and industry in Europe and the United States, including the prestigious European Research Council (ERC) Advanced Grant.