An Efficient Multi-Agent Approach to Order Picking and Robot Scheduling in a Robotic Mobile Fulfillment System

Sander Teck[a,*], Pieter Vansteenwegen[b], Reginald Dewil[a]

[a]*Department of Mechanical Engineering, Centre for Industrial Management, KU Leuven, Celestijnenlaan 300, 3001 Leuven, Belgium*

[b]*KU Leuven Institute for Mobility – CIB, KU Leuven*

Abstract

This paper presents a Multi-Agent System (MAS) for optimizing the scheduling and routing of mobile robots and human pickers in a Robotic Mobile Fulfillment System (RMFS). The RMFS is a system designed for e-commerce warehousing where autonomous mobile robots are used to fetch inventory pods, also referred to as racks, from the storage area and transport them to the appropriate picking station where human pickers pick the required number of goods. The system requires the solution of several hard decision problems like: the order-to-picking station assignment and sequencing, the pod selection, and the multi-robot task allocation. The proposed solution approach employs decentralized scheduling mechanisms, i.e. auctions and dispatching rules, to communicate and distribute picking and retrieval tasks among the agents. Various dispatching rules are identified and analyzed over a wide set of problem instances of the RMFS with varying numbers of mobile robots, picking stations, and order sizes. The proposed MAS framework shows promising results and requires only a fraction of the computation time compared to a centralized scheduling algorithm. The MAS also includes both unidirectional and bidirectional lanes. Although additional complexity in collision avoidance is introduced when using bidirectional lanes, it allows for better system performance.

Keywords: multi-agent system, distributed control, robotic mobile fulfillment system, logistics, scheduling and routing

1 Introduction

Like many other sectors, the e-commerce sector has to deal with a growing scarcity of mostly blue-collar workers. This trend has put a considerable additional strain on the whole logistics sector [1]. Automation can play a key role to alleviate this problem and take over dangerous and repetitive jobs of human workers, resulting in a significant cut in operational costs by replacing expensive human workers with ever-cheaper machines. Autonomous mobile robots (AMRs) are ideally suited to fill this gap in the labor pool as they can be deployed alongside human workers in picking processes in warehouse environments [2]. The main scheduling activities as defined by Le-Ahn and De Koster [3] are as follows: vehicle dispatching, routing, vehicle planning, deadlock resolution, battery management, and positioning of vehicles. It is important to note that these control decisions all have interdependencies.

Corresponding author *
Email address: sander.teck@kuleuven.be (S. Teck)
Postal address: Celestijnenlaan 300, 3001 Leuven, Belgium

A Robotic Mobile Fulfillment System (RMFS) is the manifestation of such a human-robot collaborative system. It is a semi-automated warehouse system that uses mobile robots for retrieval tasks in the storage areas and human workers for picking tasks in the picking area. Conventional fixed racks are replaced by movable inventory pods which are located in the storage area. One specific inventory pod can contain multiple different stock keeping units (SKUs) which allows for a reduction in required retrieval tasks if pods are selected smartly. This paper treats the assignment and sequencing of the orders to the different picking stations, the pod selection, and multi-robot task allocation problems in an automated e-commerce warehouse with rack-moving mobile robots (see Figure 1). Thus, we determine which order has to be allocated to a certain picking station, supplied with suitable inventory pods, and moved by autonomous robots in a conflict-free manner.

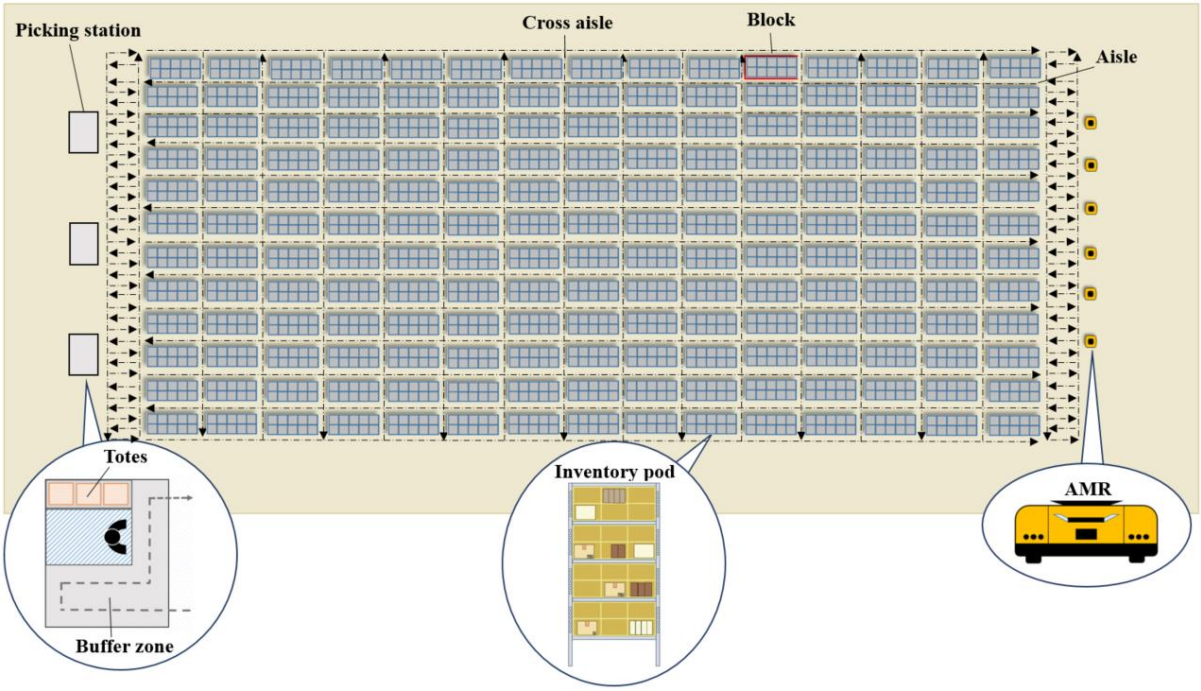

Figure 1. *Top view of a robotic mobile fulfillment system warehouse layout.*

Luo and Zhao [4] introduce a discrete-event simulation to estimate the performance of a RMFS for different layouts. Their findings show that bidirectional lanes allow for significant performance increases. However, they do not specify which dispatching/decision rules are used, and therefore, we assume that these decision rules are simple (e.g. First In, First Out (FIFO), Last In, First Out (LIFO), etc.). Merschformann et al. [5] compare some simple decision rules for the robotic mobile fulfillment system to one another. In this paper, we focus on developing more advanced dispatching rules to optimize the scheduling of AMRs and human pickers in the warehouse. We present a multi-agent-based planning approach for solving the RMFS problem. Every agent builds its own task schedule through decentralized negotiation or assignment mechanisms like dispatching rules. Multi-agent systems are very popular among decentralized systems and in some cases are even able to outperform centralized frameworks [6]. The RFMS resembles in many ways the problem considered in Erol et al. [7]. The picking stations can be viewed as production machines where the arrival times of the AMRs determine the earliest picking times at the picking stations. Vice versa, the picking stations determine the availability of an AMR to execute another retrieval task. Erol et al. [7] conclude that good synchronization and work balancing between these agents is of the utmost importance to minimize the idle time and consequently the overall operational costs. However, in the

literature, the RMFS scheduling problems are either solved with centralized algorithms which cannot be applied to very large problem instances, or through simulation frameworks utilizing very simple dispatching rules which result in low quality solutions.

In the following, we outline the main contributions of this work. We identify and validate several suitable negotiation and dispatching rules, other than the traditional dispatching rules, used by the agents in the system. We present a MAS framework for the RMFS decision problems. The framework is able to solve large-scale instances fast, while still maintaining good solution quality, whereas a centralized algorithm can take significantly longer to solve the problem. This makes the proposed MAS framework more appropriate for real-time scheduling. Moreover, we show that the proposed framework, when used in a system with bidirectional lanes, outperforms a central algorithm used on a system with unidirectional lanes for the same warehouse layout.

The remainder of this paper is structured as follows: Section 2 reviews the existing relevant literature on multi-agent-based systems and RMF systems. After that, the problem description is briefly discussed in Section 3. Next, we elaborate in Section 4 on the design of the agents within the multi-agent system and Section 5 provides a discussion of the experimental results and findings. Finally in Section 6 conclusions are drawn and future research is formulated.

2 Literature Review

The research on multi-agent task allocation is extensive and aims to (near-)optimally allocate a set of tasks $T = \{t_1, t_{2,\dots,} t_n\}$ to a set of agents (e.g. mobile robots or picking stations) $A = \{a_1, a_{2,\dots,} a_m\}$. Multi-agent systems are in fact decentralized control systems where the autonomous agents in the system coordinate and cooperate with each other in order to achieve a global goal. Applied to the RMFS, this can be the minimization of the system makespan or total distance travelled. The literature review is structured as follows: the first two paragraphs deal with relevant literature on multi-agent-based approaches and the next paragraph focuses on the literature on the RMFS. In the final two paragraphs, the literature on the multi-robot task allocation is discussed.

Erol et al. [7] developed a multi-agent-based approach to dynamically schedule both automated guided vehicles (AGV) and machines in a manufacturing setting. They proposed a system with agents using negotiation/bidding mechanisms and tested it on scheduling problems in the literature. They found that their approach outperformed the frequently used dispatching rules, which are commonly used as an alternative to bidding mechanisms, in literature. Moreover, their proposed system is competitive with some centralized optimization algorithms from the literature. The traditional decision/dispatching rules used in the literature are: First Come, First Served (FCFS), Shortest Travelling Distance (STD), Closest Available (CA), Earliest Due Date (EDD), etc. [5, 7-10]. The multi-agent architecture of this paper is based on their proposed architecture with a manager agent and an agent type for each physical resource in the system. Sahin et al. [11] also focus on the simultaneous scheduling of machines and mobile robots in a dynamic manufacturing environment. Their multi-agent-based approach which uses negotiation mechanisms, has a similar agent architecture as Erol et al. [7] and is validated with test problems from the literature. It is able to find the best known solutions and in many cases is able to improve upon them. The two previously described papers use a manager agent to initialize and create jobs, however, it can also be used to more centrally control all the agents.

In this study, it enables more advanced optimization algorithms as decision rules to perform the task allocation.

Giordani et al. [12] propose a two-level decentralized MAS framework, where at the production planning level an iterative auction based negotiation protocol is used. An update is made at each iteration for the price of assigning a robot to a task and based on these prices a utility function constructs bids, this process continues until a stop criteria is met. At the second level, the multi-robot task allocation problem is solved with a distributed version of the Hungarian Method. They compared the performance of their proposed solution approach to that of a centralized approach and found that both methods had their advantages: the decentralized method is more robust and efficient than the centralized architecture. However, the decentralized method tends to under-utilize the production resources. Furthermore, they find that the centralized policy is more effective since it uses global information resulting in lower production costs but it is characterized with a much higher robot displacement distance. De Ryck et al. [13] use a MAS for the decentral task allocation of AGVs. They propose a decentralized task allocation algorithm based on sequential single-item auctioning principles to efficiently plan the transportation tasks among the available agents. Additionally, the algorithm takes resource constraints into account during the allocation process. They compare the performance of their proposed approach to a centralized scheduling algorithm and an exact method for small scale problems. They conclude that their task allocation approach scales well with larger numbers of tasks in the system. It even outperformed the central algorithm in some cases and is able to generate competitive allocations compared to an exact approach. In this paper, we develop a MAS framework to solve large-scale problem instances for the RMFS and compare its performance to a centralized scheduling algorithm.

In the literature, several research papers focus on the operational problems of the RMFS. Boysen et al. [14] concentrate on the batching and sequencing of picking orders already assigned to a picking station. They develop an exact method and some heuristic methods to solve the problem. The study shows that an optimized pick order processing may cut the required number of robots in half to run the efficiently system compared to simple decision rules. Li et al. [15] evaluate the performance of a high-density RMFS with a simulation study and compare it to a normal RMFS layout. They conclude that a high-density RMFS layout can save up to 10% in storage area occupation. Zhuang et al. [16] consider the problem of jointly optimizing the order sequencing and the rack scheduling problem. Moreover, workload balancing and rack conflicts among multiple picking stations is also included in their optimization method. They propose an exact model and an adaptive large neighborhood search method. They conclude that up to 62% rack movements can be saved compared to company's current practice. Yang et al. [17] also consider this joint optimization problem and propose both a mixed-integer linear programming model and a two-stage solution procedure to solve it. They show that the joint optimization can result in up to 59,8% reduction of the robotic tasks compared to benchmark solutions. Valle & Beasley [18] consider the problem of allocating picking orders to picking stations and the sequencing of racks for presentation at each individual picker. They develop two matheuristics to solve the problem and prove that, under certain conditions, a feasible rack sequence can be attained just focusing on a subset of the orders to be dealt with by the picker. The paper of Teck and Dewil [19] proposes a bi-level memetic optimization algorithm to solve the integrated scheduling problem. The integrated problem consists of the order assignment and sequencing on picking stations, the pod selection,

and the multi-robot task allocation problem. The experiments show that interdependencies exist between the different sub problems which may not be disregarded during optimization. This finding is confirmed in Teck and Dewil [20] by exact models, for the sub problems and the integrated problem, illustrating the importance of these interdependencies. Furthermore, the minimization of the number of pod visits is a common objective in different research papers [14, 21]. However, in Teck & Dewil [20], it is shown that including the distance of the pod to the station in the optimization objective results in better system performance. Thus, in this study we consider the distance of a pod from the picking station when selecting pods for transportation.

In the literature, two main approaches can be distinguished to solve the multi-robot task allocation problem: the optimization-based approaches and market-based approaches [22-23].

A. *Optimization-Based Approaches*

Optimization-based approaches focus on solving problems to optimality using global information. Within this branch two main categories can be distinguished: exact approaches and heuristic approaches. Exact approaches guarantee optimal solutions. However, their usefulness quickly diminishes when the scale becomes larger since they are unable to solve them to optimality in reasonable computational time. Therefore, whenever complex combinatorial problems are concerned for medium to large-scale problems the heuristic approaches become interesting. As the name implies, the (meta)heuristic approximate approaches do not guarantee optimality but try to solve the problems to near-optimality in reasonable computational time. Zou et al. [24] developed a semi-open queueing network to compare different assignment rules, like a rule based on the handling speeds of the picking stations and a near optimal assignment rule using a neighborhood search algorithm, for the multi-robot task assignment. They demonstrate the effectiveness of their proposed assignment rules through simulation. Gharehgozli and Zaerpour [25] focus on the multi-robot task assignment problem to fulfill the picking orders at the picking stations in a RMFS. They model the problem as an asymmetric traveling salesperson problem and extend it with general precedence constraints. Moreover, they propose a heuristic method to solve large size instances. However, their work assumes the order-to-picking station assignment as predetermined. The suitability of optimization based approaches in multi-robot task allocation highly depends on the optimization time that is available to solve the problem since they generally take longer than market-based approaches. In this paper, we identify and propose some optimization-based dispatching rules to efficiently handle the multi-robot task assignment problem.

B. *Market-Based Approaches*

Market-based, commonly referred to as auction-based approaches are, compared to the optimization-based approaches, very scalable and flexible. The auctions are a way to perform the resource allocation in a multi-agent system. Overall, the auction principles can be divided into three different methods: the parallel single-item auctions, the combinatorial auctions [26, 27], and the sequential single-item auctions [13]. Parallel single-item auctions, in contrast to sequential single-item and combinatorial auctions, allocate various different items (e.g. orders and tasks) simultaneously. As a result, it disallows the bidders to take interdependencies between these items into account. Therefore, the computational complexity of parallel single-

item auctions ($O(n_{bidders} * m_{items})$) is significantly lower than that of sequential single-item auctions ($O(n_{bidders} * m^2{}_{items})$) and combinatorial auctions ($O(n_{bidders} * 2^{m_{items}})$). However, the parallel single-item auction does not guarantee good quality, whereas combinatorial auctions can reach close to optimal solutions [13]. The advantage of the sequential single-item auction mechanism is that it is a compromise with regards to the computational complexity and the solution quality between a parallel and a combinatorial auctioning mechanism [13]. The solution quality is typically better than the parallel auctioning mechanism because sequentially auctioning items allow the bidders to take synergies between the items into account in their bid calculation. We develop a parallel single-item auctioning mechanism to distribute the picking orders and transportation tasks over the available agents and compare its performance to the proposed optimization-based approaches.

In this study we consider several decision problems of the RMFS: assigning the orders to the different stations and sequencing the orders, selecting the pods, and allocating the multi-robot tasks. A previous paper on this integrated problem [19] proposes a centralized optimization algorithm. Even though it is able to obtain good quality solutions, it requires too much computation time to be readily applicable for real-time scheduling purposes. In this work, we propose a multi-agent approach that is able to solve this problem in a fraction of the computation time that the memetic algorithm requires. Furthermore, it is able to solve larger problem instances which we also introduce in this paper.

3 Problem description

We consider several sub problems of the RMFS. The order-to-station assignment and sequencing problem concerns how to allocate orders from a set of available orders to picking stations and when. This in a way that minimizes the makespan of the picking stations and maintains a good work balance. Specific to the RMFS, is the capability to simultaneously work on multiple orders at the same time on the workbenches of the picking stations. Whenever orders have been assigned to picking stations, the picking stations seek to fulfill these orders by looking for suitable inventory pods. This is called the pod selection problem. A typical inventory pod contains multiple different SKUs. In a RMFS a scattered storage policy is typically applied [28], meaning that when an order has to be completed it can potentially be completed by various different inventory pods. However, in many cases there is one inventory pod that is more suitable than the others. This is mainly dependent on the location of the pod to the picking station that can serve the specific order line. Moreover, consolidation opportunities in the fulfillment of orders at the picking stations is an important consideration during the process of selecting the pods. In a previous paper, we introduced the term of 'pod consolidation' which refers to the average number of order lines that can be fulfilled by an inventory pod when it is presented to a human worker at a picking station [19]. Once suitable inventory pods have been selected, pod retrieval tasks can be distributed among the mobile robots. The multi-robot task assignment problem deals with the problem how to assign these retrieval tasks the different mobile robots. The assignment of these task have to be in such a way that the total distance travelled is minimized. A characteristic unique to this system is that an inventory pod can directly be transported from one station to another. Furthermore, in the literature, unidirectional lanes are commonly used since these allow for the simplifying assumption that robot congestion or deadlocks will not occur during operation. This assumption is reasonable and relates closely to the real situation [29]. However, bidirectional lanes may

allow for better system performance. In this paper, both unidirectional and bidirectional lanes are tested.

For a more detailed description of the operations in a RMFS we refer to the paper of Enright and Wurman [30]. The performance criteria in this study are the time at which the last order at a picking station is fulfilled, defined as the makespan and the total distance travelled by all robots. These are typical metrics to compare the performance of the RMFS. However, in reality these two metrics have different weights. In Teck & Dewil [20] we introduce one cost metric where we determine an hourly wage cost (c_{wage}) of €18.75/hr for one human worker at a picking station (PS) and a cost of operating a robot ($c_{distance}$) per driven kilometer of €1.75/km. The proposed metric includes the makespan in hours (t_m) and distance in kilometers (d_{total}), with their respective weights, as a cost. This is the objective that we optimize in this study and looks as follows:

$$Cost = PS_{available} \cdot t_m \cdot c_{wage} + d_{total} \cdot c_{distance} \quad (1)$$

The main assumptions made in this paper are the following:

I. Pods are fully stocked from the start time of the warehouse operations and the units are assumed to be inexhaustible, meaning that the pods always contain sufficient units to satisfy the requested order-lines.
II. The inventory pods have a fixed position in the storage area.
III. Robot breakdown does not occur and battery management is not included.
IV. Robots travel at a constant velocity.
V. A robot will dwell at the location where it completed its latest service if no other task is immediately assigned.
VI. Robots always travel to their destination following the shortest path and unloaded robots can travel underneath the inventory pods in the storage area.
VII. Robots are not dedicated to one specific picking station, but are pooled over all stations.
VIII. Picking times are deterministic.

4 The proposed multi-agent-based framework

In this work, we opted for an event-driven simulation approach since it has several benefits over a more traditional time-driven approach. First of all, event-driven simulation is more efficient, since it only regards time steps where actual changes (events) occur. This results in less computation effort because only the agents that are affected by the specific events have to be updated. Secondly, it allows a flexible and varying level of complexity of the model [31]. Figure 2 illustrates the high-level control flow chart of the discrete-event simulation (DES) environment. After the initialization of the environment and its agents, the status of the Pickingstation Agents is checked after picking events occur. If new slots open up on a workbench, new order totes can be assigned to the available Pickingstation Agents. Thereafter, the assignment processes occur as described in the following sections. In the case of no new open slot, the simulation time advances. This process continues until all customer orders have been fulfilled.

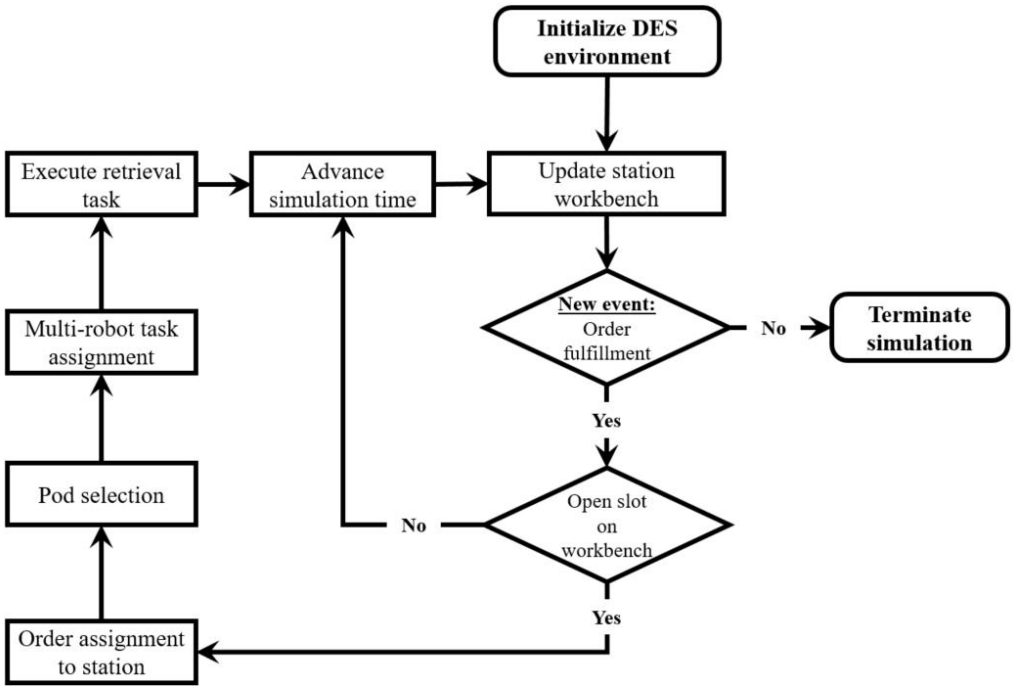

Figure 2. *Control flow chart of the DES environment.*

The agents in this framework are structured using object-oriented concepts. In this MAS implementation, all resources are modeled as agent types where each agent type has its own specific behavior and capabilities. In the remainder of the paper we refer to the agent controlling a vehicle as a Vehicle Agent, the agent responsible for a picking station as a Pickstation Agent, the agent for an inventory pod as a Pod Agent, and the agent controlling the allocation of an order to the picking stations as an Order Agent. In figure 3, the proposed multi-agent architecture is shown. The architecture consists of one Manager Agent and multiple local agents, further described in the next section. All the local agents, which represent physical resources, can consult and use the information of the Manager Agent and vice versa. The local agents can only interact with specific agents, for instance, an Order Agent can only interact with the Pickstation Agents, while Pickstation Agents can also interact with Pod Agents and Vehicle Agents.

Table 1. *Nomenclature.*

Symbolic expression	Description
$p_{assigned}$	Orders already assigned to picking stations
$p_{auction}$	Orders being auctioned
p_{total}	The total amount of orders
d_{pod}	Travel distance from current location to pod location
t_{dwell}	Estimated time required to travel from dwell to new pod location
$t_{completion}$	Estimated time to complete last retrieval task in the queue
c_t^i	Marginal cost of assigning a retrieval task to a mobile robot where the index i refers to the ranking (i.e. i = 1 is the best marginal cost)
O	The set of customer orders
N	The set of all retrieval tasks in the system
AMR	The set of available autonomous mobile robots
PS	The set of available picking stations
X	The dimensions of the look-ahead tree in both width and depth
T^A	The set of retrieval task that have to be assigned at that time to AMRs

4.1 Agent definition

In the proposed multi-agent-based framework the agents are determined based on their functions and goals in the system. We are defining the various types of agents operating in the warehouse system as follows.

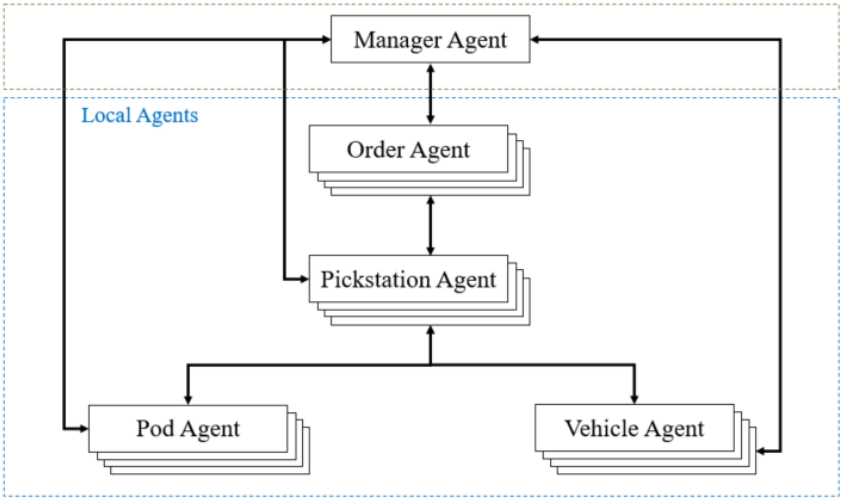

Figure 3. *The proposed multi-agent architecture.*

4.1.1 Manager Agent

In the proposed multi-agent architecture there exist only one Manager Agent and it has a supervisory role. It has two main functions in the system: the initiation of the system and the creation of all the resource agents and the assistance of these resource agents (e.g. order, picking station, vehicles, and inventory pods) during operation. The number of resource agents available in the system is determined beforehand by the decision maker. Furthermore, since the Manager Agent has access to the information of all available local agents it enables the use of more centralized algorithms (see Section 4.2.4) for the decision making process. Therefore, the Manager Agent is able to influence the decisions made by the local agents.

4.1.2 Order Agent

The Order Agent contains all the details concerning the order (e.g. order lines, stock keeping units (SKUs) required, etc.) and its status. Thus, for every order an Order Agent will be created. It keeps track whether the picking order is already being processed at a picking station or if it is still available for allocation in the backlog. Furthermore, the Pickstation Agents have to decide which new order can become active at their workbench. They can check the consolidation opportunities of a given order through its Order Agent.

4.1.3 Pickstation Agent

During the operation of the system, the picking stations are responsible for handling the incoming orders. Each picking station has an order capacity, basically limiting the number of orders that can be actively worked on at a certain time at the workbench. Therefore, it is highly important for the picking stations to determine which orders have to be fulfilled first and in which sequence. Whenever an order has been completed, it is removed from the list of active orders and the Order Agent will change its status to ‘Completed’. Thereafter, another order can become active, in this case the picking station will check the list of unallocated orders that still have to be completed. It will look for an order that complements its current list of active orders best, based on consolidation opportunities. Next, it requests from the respective Order Agent

to initiate an auction for it. An auction is initiated because other picking stations may also be suited for that specific picking order. Therefore, the competing picking stations compute a bid value to participate in the auction. A Pickstation Agent not only has the role of a bidder, but can also take the role of auctioneer. For instance, when it makes requests for inventory pods to fulfill the newly arrived order lines and also in the situation where it assigns transportation tasks to the available mobile robots. Thus, when it takes the role of auctioneer it is responsible for both the announcement of the auction, the winner determination problem, and the winner announcement to all the participants of the auction (see Figure 4). The overall objective of the Pickstation Agents is to complete all the arriving orders in the shortest makespan possible. This assumes that all human workers remain in the warehouse until the last picking task is completed.

4.1.4 Pod Agent

The role of an inventory pod is to supply the picking stations with SKUs in order to fulfill customer orders. The distance between an inventory pod and a picking station has a significant impact on the overall travelled distance, therefore, one of the objectives is to minimize this distance. In Section 4.2.1, the allocation process for order lines-to-inventory pods is shown.

4.1.5 Vehicle Agent

In the RMFS, mobile robots have to transport inventory pods from the storage area to the picking stations that require them and back. They have the capability to request tasks from the Pickstation Agents in case their current task queue is empty, this to maximize their uptime. Whenever a Pickstation Agent announces an auction for a transportation task, the Vehicle Agent has to compute a bid value proportional to its eagerness to execute the retrieval task. The Vehicle Agent computes its bid taking into account the vehicle's estimated time when it is available for a new task and the travel time from its planned dwell location to the newly requested pod location. In Section 4.2, several different decision rules are proposed to allocate the transportation tasks to the AMRs. The objective of the Vehicle Agents is to perform all the transportation tasks travelling the shortest distances possible in a collision and deadlock free manner.

4.2 Decision rules and negotiation mechanisms

In this study several negotiation mechanisms and decision rules were developed to distribute the orders to the picking stations and retrieval tasks to the mobile robots. These mechanisms or rules can be completely decentralized or entirely centralized through the use of the Manager Agent to coordinate the other involved agents. A commonly used simple dispatching rule (FIFO), is also used in the experimental section to compare the proposed mechanisms to.

4.2.1 Sequential single-item auction mechanism

In the decentral sequential single-item auction (SSIA) mechanism, an unallocated order/task is presented to the interested bidders simultaneously. Every bidder has the opportunity to propose a bid based on their intentions for the specific order/task. The auction process typically goes as follows: an auctioneer initiates an auction for a single order/task and announces the start of the auction to all the potential bidders. Next, it awaits for the bidders to compute and propose their bids. Thereafter, the auctioneer has to decide which bid is the best. The winning bidder receives the order/task and all the other bidders are also informed of the results of the auction. Both the picking stations, the mobile robots, and the inventory pods use this mechanism to allocate either picking orders or retrieval tasks respectively. Consider the first case where an Order Agent

initiates an auction to allocate a picking order to a picking station. The Pickstation Agents with the highest bids wins. They calculate their bid (2) based on the consolidation gain, expressed as the number of pod visits saved. For instance, if the order has three order lines where two of the three order lines can be fulfilled by the inventory pods requested for the other current active orders on the workbench, then potentially two pod visits have been saved. The work balance is also factored in the bid calculation. This work balance is expressed as the number of picks that have to be performed to fulfill all assigned orders ($p_{assigned}$) and the order that is being auctioned ($p_{auction}$) divided by the total number of picks (p_{total}) that have to be performed over all stations. The workload has a negative impact on the bid value in order to maintain the work balance over all stations.

$$Bid = consolidation \cdot (1 - (p_{assigned} + p_{auction})/p_{total}) \quad (2)$$

Consider the second case, the auctioning process for the pod selection problem. A Pickstation Agent requests an inventory pod for the fulfillment of a line of one of its orders and initiates an auction. Only the Pod Agents that represent an inventory pod that contain the required SKUs can participate in the auction. As mentioned before, the main consideration in the bid calculation is the travel distance (d_{pod}). However, since one inventory pod contains multiple SKUs there is a possibility that multiple of those SKUs are required by either one or multiple picking stations. In this case it might be more desirable to have an inventory pod that is located further away but can fulfill multiple order lines in one visit since the average distance per order line can be shorter [20]. Thus, this consolidation is included by dividing the distance over the number of order lines that can be consolidated, the lowest bid wins (3).

$$Bid = d_{pod} \,/\, consolidation \quad (3)$$

There are situations where picking orders become active that require SKUs that are stored on inventory pods which are already assigned or even being transported to the picking stations. In such a case those SKUs are not taken into account in the auctioning process. As illustrated in Figure 4, a Pickstation Agent initiates an auction to assign a new retrieval task to a mobile robot. The involved Vehicle Agents make their bid calculations (4) based on the expected time of arrival, the lowest bid wins. Thus, they include the time at which they expect to complete their last retrieval task in the queue ($t_{completion}$) and the time required to travel from the dwell position to the new inventory pod location (t_{dwell}).

$$Bid = t_{completion} + t_{dwell} \quad (4)$$

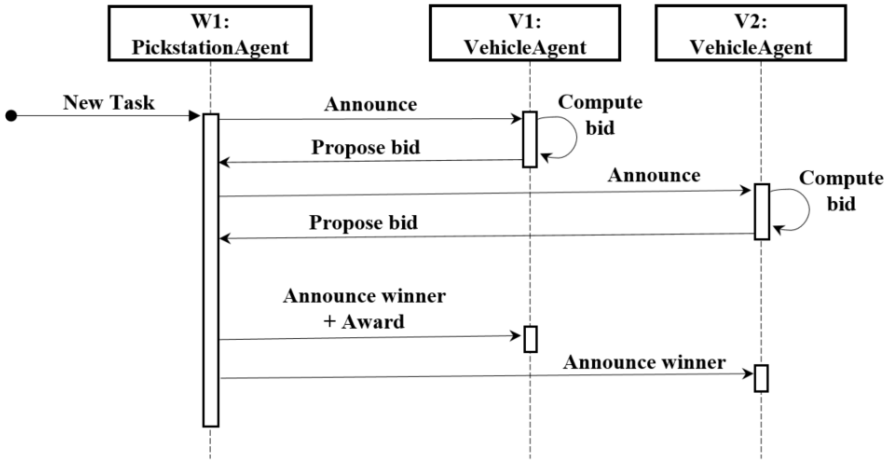

Figure 4. *UML interaction diagram of an auction procedure with one picking station and two vehicles.*

4.2.2 Greedy look-ahead heuristic

The main downside of the simple sequential single-item auction mechanism is that it does not account for the impact of assigning a certain pod retrieval task to a mobile robot can have on other mobile robots. Allocating a retrieval task to a less suitable mobile robot can result in an overall longer distance travelled by all the mobile robots and a longer system makespan. Therefore, we propose a look-ahead heuristic (see Algorithm 1) where a decision tree is constructed for the multi-robot task assignment. The tree attempts to foresee the described impact of selecting a certain retrieval task for assignment on the overall distance travelled and the system makespan. This heuristic is executed by the Manager Agent who communicates with the Vehicle Agents to determine the bidding values. The look-ahead and evaluation procedure, used to find the most promising tree nodes, is inspired by the heuristic proposed in Jin et al. [32]. Our heuristic differentiates itself by partially looking ahead and evaluating these options, from these options initiate another look-ahead step and so on. More specifically, the procedure initially allocates a retrieval task to an available vehicle following the same principles as the sequential single-item auction and evaluates the impact on the makespan of all the available vehicles. The procedure does this for every single task in the set of unallocated tasks. Once every retrieval task, in the set of unallocated tasks, has been evaluated, the heuristic continues only with the X best scoring options.

Algorithm 1: Look-ahead Procedure

$assign_{task} \leftarrow \emptyset$, $tree_{dim} \leftarrow X$ *(tree dimensions)*
Initialize tree nodes for retrieval tasks $t_n \; \forall \, n \in N$ *(where N is the set of tasks)*
For t_n in N
    node ← score for t_n based on formula (4) for each Vehicle Agent
$promising\_branches \leftarrow$ select $tree_{dim}$ most promising task assignments *(Evaluation)*
$count \leftarrow 1$
While $count \leq tree_{dim}$
    For *branch* in *promising_branches*
        For t_n in N | $t_n \notin branch$ *(find score for all tasks that are not already assigned in the branch)*
            node ← score for t_n based on formula (4) for each Vehicle Agent
    $promising\_branches \leftarrow$ select most promising node in each branch to continue the branch *(Evaluation)*
    $count + 1$
End
$assign_{task} \leftarrow$ select origin node of the best *branch* in *promising_branches*
return $assign_{task}$

In Figure 5, an example is given of the construction of the tree, where X determines both the tree depth and width. In this case, the X is three. Thus, the tree search is limited in both search depth and width and expands the tree only with the three most promising nodes. If one would consider all branches instead of just X branches, an exhaustive search is performed and as the number of tree nodes increases exponentially as the tree depth increases, it becomes too time consuming. Starting from the root node, according to the depth and width limitations a limited look-ahead tree is constructed. Finally, when the limited look-ahead tree is constructed, the leaf node with the best makespan over all vehicles is selected as the best branch. The initial task from this branch has been chosen as the most suitable task to be allocated to a vehicle. Thus, each time the greedy look-ahead heuristic is called, a single task is allocated.

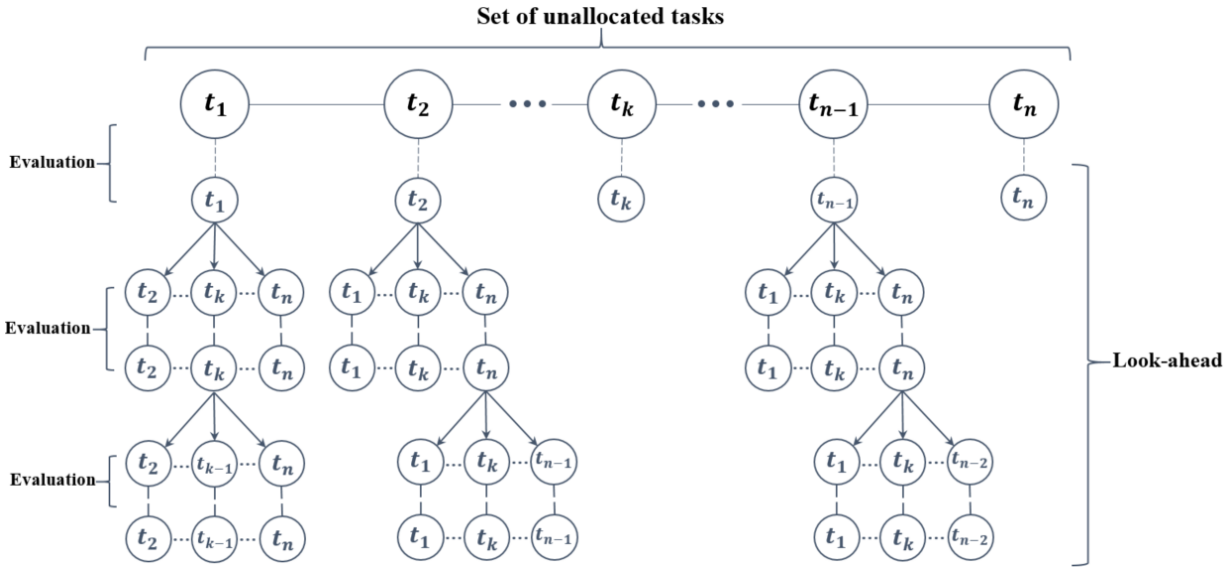

Figure 5. *Greedy look-ahead heuristic.*

4.2.3 Regret-based task selection

This regret-based approach for the task selection also provides a form of look-ahead information in order to iteratively assign a retrieval task to a mobile robot. The regret heuristic is also a more centralized algorithm available through the Manager Agent. This procedure selects tasks that will result in the least regret later on compared to the situation where you would not select this task now. The task with the highest regret value is selected to achieve this result. The regret value is based on the difference of assigning the task to the mobile robot with the lowest marginal cost c_t^1 and the mobile robot with the second-best marginal cost c_t^2 [33-34] (Potvin & Rousseau, 1993; Chen et al., 2021). However, the look-ahead information can be extended by taking more alternatives into account and not only the marginal cost of the best and second-best option. Hence, it can also include the difference between the best and third-best alternative and so on. Here, the k in the regret-k heuristic determines the number of alternatives that are included in the regret calculation. For each task t in a set T^A of tasks that are to be allocated to picking stations the selection of a task t happens as follows [35]:

$$t = max_{t \in T^A}\left(\sum_{h=2}^{k} c_t^h - c_t^1\right) \quad (5)$$

4.2.4 Lin-Kernighan-Helsgaun

The main drawback of the previous allocation mechanisms is that they mainly work on local information. In contrast, centralized heuristics, like the Lin-Kernighan-Helsgaun heuristic, can use global information. This method simultaneously assigns multiple tasks to the mobile robots. This has as a main benefit that the routes of the mobile robots will be of very good quality. However, this comes at the cost of more computational effort to generate routes. It is important to note that when the routes are being optimized the pod selection for the current active orders has already been determined.

The state-of-the-art LKH-3 algorithm is implemented as a task allocation method for solving vehicle routing problems [36]. This algorithm is an extension of the classical Lin-Kernighan heuristic for solving the well-known travelling salesperson problem [37] adapted to be applicable to the vehicle routing problem. The LKH-3 heuristic uses penalty functions for handling constraints. It can be called by any Vehicle Agent through the Manager Agent, since it requires a more global view of the system to provide the required input data to run the solver. For this specific use case, it was chosen to solve the problem as an open vehicle routing problem (OVRP). The LKH-3 models the pod retrievals from the pod locations to the picking stations and back performed as well as the movement of the mobile robots from their dwell location to the next pod retrieval location. The objective is to minimize the overall distance travelled. As the sub problem being solved only deals with a limited number of retrieval tasks after which additional jobs still need to be executed, there is no requirement for the mobile robots to return to their origin locations and, therefore, an OVRP as a model is justified. Each time the situation in the system changes, we re-optimize the task retrieval sequences of the mobile vehicles.

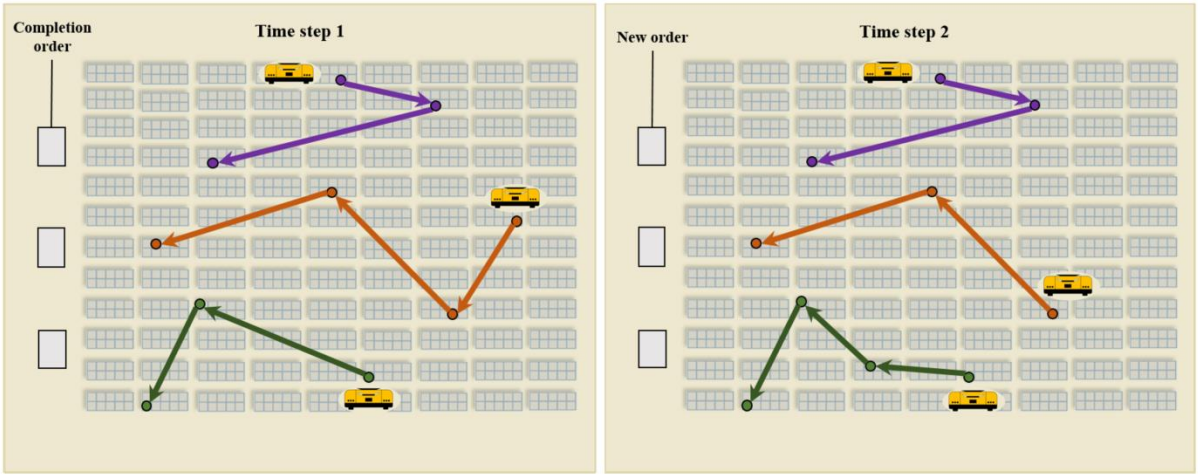

Figure 6. *LKH-3 re-optimization after new order arrival. The routes in the figure do not illustrate the actual paths followed by the AMRs but show in which sequence the AMRs will retrieve the pod requests.*

In Figure 6, the vehicle following the orange sequence finished its first retrieval task, in time step 1, in the sequence resulting in an order completion and a new order arrival in 2. Thereafter, the LKH-3 heuristic re-optimizes the task retrieval sequences given the initial sequence and the newly arrived retrieval request(s). The starting locations correspond to the planned dwell locations of the mobile robots after they have finished the retrieval task that they are executing when the solver has started optimizing. After re-optimization, the vehicle following the green sequence is assigned a new retrieval task in its sequence. The multi-agent system continues until all pick orders have been fulfilled

4.3 Conflict-free routing considerations

In most cases, the dispatching, routing, and scheduling are not as straightforward as just finding the shortest paths between origin and destination. In theory, all operations can be completed just as planned with no conflicts. In the literature, the assumption of unidirectional lanes is commonly made [29]. However, when bidirectional lanes are concerned it is important to note that in such a case conflicts may arise during operation and these cannot be disregarded during the scheduling. Some common conflicts are: vehicle collisions, congestion, livelocks, and deadlocks. Livelocks and deadlocks occur in situations where vehicles cannot move anymore due to mutual blocking. Therefore, collision avoidance and deadlock resolution are important considerations during modelling [38-39]. In the developed multi-agent system a conflict-free routing mechanism is implemented for the Vehicle Agents. This mechanism consists of three phases (see Algorithm 2): the individual path planning (Phase 1), collision detection (Phase 2), and conflict resolution (Phase 3).

Algorithm 2: Collision Avoidance Strategy

Before simulation execution

1 Compute shortest path matrix *(Floyd Warshall algorithm)*

2 ***Initialize*** $vehicle_a \; \forall \; a \in AMR$ *(where AMR is the set of mobile robots)*

During simulation execution

3 $route_1 \leftarrow \emptyset$ *(initially no route for* $vehicle_1$*)*

4 $vehicle_1$ requests to execute a retrieval task

5 $route_1 \leftarrow$ retrieve route with shortest path matrix *(Phase 1)*

6 **While** $route_1$ in conflict with any $route_a \; \forall \; a \in AMR$ *(Phase 2)*

7 $conflict_1 \leftarrow$ conflict determination *(Figure 6)*

8 **If** $conflict_1$ is a cross collision

9 $vehicle_1$ waits until conflict is resolved *(Phase 3)*

10 **Else**

11 $route_1 \leftarrow$ modified A* algorithm *(Phase 3)*

12 ***Execute*** $route_1$

In Phase 1 a Vehicle Agent constructs a path from its current position to the destination (rule 5) with a pre-generated shortest path matrix (rule 1). This allows to quickly generate the shortest path. Next, in Phase 2, the Vehicle Agent employs the collision detection algorithm to find potential conflicts in its current planned route with the routes of the other Vehicle Agents in the system (rule 6). Since a route consists of a list of nodes that will be visited, the detection algorithm can calculate the position and time of each vehicle in the system throughout the routes. Hence, the algorithm checks for each time step in the new proposed route whether the vehicle would occupy the same node with another vehicle that might already be executing its route. Therefore, if two or more vehicles occupy the same node at the same time step, this results in a conflict and the conflict type can be determined (rule 7). If no conflict occurs, the originally planned routes can be followed. Otherwise, the conflict has to be resolved in Phase 3 (rule 9 and 11). The conflict resolution algorithm, based on the conflict type that occurred, either applies a waiting strategy or a rerouting strategy. In Figure 7, the most common conflicts are shown. Both 7a, 7b, and 7c represent a cross collision, 7a and 7b denote that the mobile robots will travel along different routes as they pass the intersection and in 7c the mobile robots will travel on the same route after passing the intersection. In 7d the mobile robots want to pass through the same aisle. Conflicts 7a, 7b, and 7c can be resolved using a waiting strategy where one of the mobile robots in conflict can be instructed to wait for a certain period until there is no conflict anymore. Conflict 7d has to be resolved with the rerouting strategy. If rerouting is required, a modified A* algorithm partially reconstructs a new route avoiding the conflict area. Incorporating conflict-free routing mechanisms is also necessary if bidirectional lanes are used.

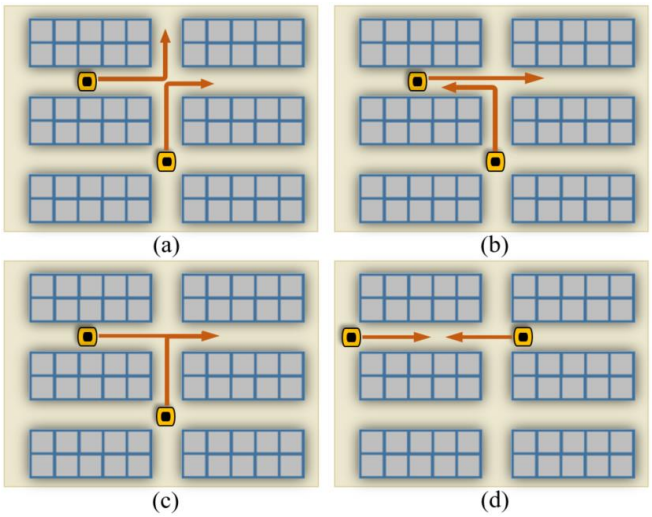

Figure 7. *Common routing conflicts: (a), (b) and (c) denote cross collisions and (d) illustrates the robots wanting to use the same aisle.*

5 Computational experiments

In this section, we validate the MAS approach and compare its performance to the optimization-based solution. We generated several instances over a range of problem types as follows: instances[1] range in size from 50 to 1000 orders (N), the available number of autonomous mobile robots ranging from 3 to 48 (AMR), and 3 to 12 picking stations (PS). The orders used in the problem instances contain an average of two order-lines. The majority of the orders only have 1 or 2 order-lines. A truncated exponential distribution is used to generate the number of lines per order. The general warehouse layout of all instances used in this paper are presented in Table 2 and are based on the configurations used in the tactical decision making study of Lamballais et al. [29] focusing on the impact of the warehouse layout on the system performance.

Table 2. Characteristics used to generate the warehouse layout.

Characteristic	Value
Number of vertical shelves in a block	2
Number of horizontal shelves in a block	5
Number of aisles	17
Number of cross-aisles	9
Number of shelves	1800
Length of side aisle	5m
Unit length	1m
Number of robots	3, 6, 12, 24, 48
Robot velocity	1.3 m/s
Time for pod lifting and storing	1s
Number of picking stations	3, 6, 12
Buffer of picking stations	5
Station order capacity	5
Picking time	10s

[1] https://www.mech.kuleuven.be/en/cib/rmfs

In all experiments performed, the optimization metrics are the makespan of the overall system and the total distance travelled by the mobile robots. The proposed multi-agent framework is programmed in Python 3.8 and the instances were run on a device with the following specifications: Intel Core i7-9850H CPU @2.60GHz, installed 32GB RAM, and 6 cores (12 logical processors).

5.1 Problem instances with unidirectional lanes

For this experiment, we consider unidirectional lanes which allows us to not have to consider congestion and deadlock avoidance [4, 18, 19, 24, 25, 29]. This assumption relates closely to reality since single directional lanes and the fact that unloaded robots can travel underneath the pods in the storage area strongly decreases the occurrences of congestion and deadlocks. We validate and compare the approaches proposed in Section 3 and compare their performances to a centralized bi-level memetic (MA) optimization approach [19] and a dispatching algorithm proposed by Qin et al. [40]. For the MA approach we use a population size of 34, a mutation probability of 0,092, an elite size of 4 individuals, and a stopping criterion that is either based on the number of iterations without improvement exceeding 26 iterations or a time limit of 1 hour. All these setup parameters where chosen after a systematic parameter calibration. A fractional factorial design was chosen to restrict the number of required experimental runs but still be able to determine possible interaction effects between the parameters. The calibration was used on a subset of the problem types in order to get parameter setting that obtain good quality solution for the complete range of problem instances. Furthermore, the objective function of the MA is the same as the optimization objective discussed in (1). Qin et al. [40] developed a dispatching algorithm (integer program model) to make real-time dispatching decisions among robots, rack, and picking stations in a RMFS. They implemented their method in a real-world e-commerce company and improved its performance significantly. Each operating period of three seconds, the matching problem between the AMRs, inventory pods, and picking stations has to be solved. However, they consider the order-to-picking station allocation as given. We implement their dispatching approach in our simulation model to compare it with our proposed approaches. Therefore, to have a fair comparison, the order allocation method is the same as for all our proposed dispatching methods (see Section 4.2.1) excluding the FIFO rule.

We simulate 3 to 48 mobile robots in a warehouse layout with 3 to 12 picking stations and orders ranging from 50 up to 1000 for each experimental setup, as stated in Section 4. Table 3 shows the optimization metric, expressed as a cost, for the different developed approaches. Remark that we are not able to find optimal solutions for this complex integrated problem [20]. This was expected since Boysen et al. [14] were unable to solve instances larger than 10 orders to optimality within the time limit of 1 hour for one of the sub problems of the integrated problem. Therefore, we opted for a centralized scheduling approach which can obtain good quality solutions. From the results it can be seen that the memetic algorithm has the best performance in all the experimental setups on both the makespan and total distance travelled metrics. These results will be used as a benchmark to compare with the results of the proposed MAS framework to determine the performance gap between them.

First of all, the worst performing task allocation mechanism for the multi-agent system, with an average gap of 37,2%, is the MAS_{FIFO} decision rule. This was to be expected since the agents in the system do not negotiate and coordinate with one another to optimize their operations and

just fulfill order retrieval tasks following a simple decision rule. When the agents negotiate more with one another, for instance through the sequential single-item auctioning method (MAS_{SSIA}), the system performance increases. However, this method, with an average gap of 17,8%, performs worse than the allocation mechanisms that can evaluate the future impact of a decision. The look-ahead heuristic (MAS_{LA}) with its limited tree search approach keeps the computational load manageable. However, some potential good solutions can be missed during the look-ahead search. With an average gap of 16,1% it performs worse than the regret-k heuristic (MAS_{regret}) with an average gap of 5,2%. The regret-k heuristic can be seen as an implicit look-ahead approach and is able to prevent situations where certain tasks have to be allocated to ill-suited mobile robots because the better suited mobile robots are no longer available. From preliminary experimental results we found that a k-value equal to the number of AMRs in the system worked best for the set of instances. Furthermore, with an average gap of 14%, the approach of Qin et al. [40] is outperformed by MAS_{regret}. We believe this is because their integer program, when assigning a transportation task, does not consider the impact on the future assignment of transportation tasks. This is what the regret-k and LKH-3 heuristic do better. Finally, the task allocation mechanism that achieves the best solution quality is the LKH-3 heuristic (MAS_{LKH-3}) with an average gap of 2,0%. As expected, whenever more global information is included in the task allocation process, the overall performance of the system can be more optimized. However, this use of more global information during the optimization process results in more computation time. Therefore, the regret-k heuristic might be more suitable for application in real-world systems since it has a good balance between solution quality and computation time.

Table 3. *Simulation results with unidirectional lanes comparing the proposed mechanisms to the centralized algorithm (MA). The costs used for comparison are based on the calculation in (1).*

O/PS/AMR	MA [19]	FIFO	Gap	SSIA	Gap	Look-Ahead	Gap	Qin et al. [40]	Gap	Regret-k	Gap	LKH	Gap
50/3/3	51.1	69.7	-36%	62.9	-23%	57.6	-13%	57.2	-12%	56.0	-10%	53.7	-5%
50/3/6	32.1	45.7	-42%	39.3	-22%	37.7	-17%	36.3	-13%	37.6	-17%	34.6	-8%
50/3/12	24.7	33.9	-37%	30.3	-22%	27.5	-11%	27.3	-11%	28.0	-13%	27.0	-9%
100/3/3	72.7	105.5	-45%	98.8	-36%	90.9	-25%	85.7	-18%	81.3	-12%	86.4	-19%
100/3/6	47.4	69.0	-46%	59.6	-26%	56.6	-19%	54.4	-15%	53.8	-14%	52.6	-11%
100/3/12	33.1	48.1	-45%	41.7	-26%	41.7	-26%	44.0	-33%	40.1	-21%	39.4	-19%
200/3/3	180.7	249.3	-38%	203.3	-13%	206.3	-14%	200.2	-11%	185.0	-2%	182.4	-1%
200/3/6	113.4	158.0	-39%	125.1	-10%	130.9	-15%	131.4	-16%	119.6	-6%	118.1	-4%
200/3/12	80.8	113.9	-41%	107.0	-33%	98.3	-22%	98.4	-22%	87.4	-8%	88.2	-9%
200/6/6	175.3	251.5	-44%	207.6	-18%	209.6	-20%	193.4	-10%	190.8	-9%	183.8	-5%
200/6/12	113.8	162.9	-43%	130.3	-15%	135.5	-19%	133.6	-17%	125.0	-10%	117.7	-3%
200/6/24	80.2	117.5	-46%	96.4	-20%	94.1	-17%	99.1	-24%	88.9	-11%	88.1	-10%
500/6/6	424.3	576.0	-36%	415.0	2%	469.7	-11%	474.2	-12%	433.2	-2%	416.3	2%
500/6/12	259.9	350.6	-35%	303.5	-17%	289.6	-11%	299.0	-15%	254.3	2%	260.9	0%
500/6/24	193.1	259.4	-34%	227.9	-18%	224.8	-16%	226.9	-18%	197.4	-2%	185.5	4%
500/12/12	399.4	540.9	-35%	474.0	-19%	470.9	-18%	441.1	-10%	413.0	-3%	397.3	1%
500/12/24	269.4	348.1	-29%	300.8	-12%	312.4	-16%	292.7	-9%	268.2	0%	261.9	3%
500/12/48	202.7	258.4	-28%	223.8	-10%	223.6	-10%	222.0	-10%	198.2	2%	196.7	3%
1000/6/6	873.6	1147.6	-31%	785.0	10%	983.0	-13%	915.9	-5%	860.1	2%	812.5	7%
1000/6/12	533.2	729.1	-37%	738.9	-39%	614.0	-15%	596.9	-12%	526.2	1%	496.3	7%
1000/6/24	373.4	525.7	-41%	491.8	-32%	478.0	-28%	442.0	-18%	368.9	1%	363.9	3%
1000/12/12	852.8	1121.6	-32%	935.7	-10%	972.7	-14%	952.6	-12%	860.0	-1%	828.5	3%
1000/12/24	570.4	711.6	-25%	610.3	-7%	592.3	-4%	604.4	-6%	539.9	5%	496.6	13%

1000/12/48	403.1	507.7	-26%	456.8	-13%	450.0	-12%	434.9	-8%	395.0	2%	357.0	11%
Avg. gap			**-37.2%**		**-17.8%**		**-16.1%**		**-14.0%**		**-5.2%**		**-2.0%**

Table 3 shows the computational time in function of the number of orders to be fulfilled in a RMFS. Note that the computation time of the memetic algorithm for the larger order sizes is cut off at 1 hour of runtime. This cut off also explains why the memetic algorithm starts to perform poorly for large scale instances since it is unable to find very good solution within this time limit. From Table 4, we can see that the look-ahead and LKH-3 heuristics are considerably slower than the remaining three approaches. The reason for this is that these two allocation methods use more global information to allocate retrieval tasks to mobile robots and require multiple iterations to improve the solution quality. The LKH-3 iteratively optimizes its solution with r-opt moves, while the look-ahead heuristic estimates the impact of a decision now on the situation several iterations in the future. The remaining decision rules and allocation mechanisms require just a single constant time complexity calculation to make an assignment, hence, making them less computationally expensive. The regret-k heuristic is able to find good quality solution in only a fraction of the time required by the best performing allocation heuristic and the memetic algorithm. Furthermore, it is interesting to note that scheduling a certain amount of orders takes significantly longer when more resources (AMRS and picking stations) are involved. The reason for this is that there are more decision options when more resources are concerned, hence, it requires more computation time from the heuristics.

Table 4: *Computation time comparison for the different task allocation mechanisms.*

O/PS/AMR	FIFO Time [s]	SSIA Time [s]	Look-Ahead Time [s]	Regret-k Time [s]	LKH Time [s]	Qin et al. [40] Time [s]	MA [19] Time [s]
50/3/3	0.33	0.32	1.90	0.39	9.79	2.20	56,04
50/3/6	0.35	0.33	2.72	0.39	14.17	1.91	57,73
50/3/12	0.34	0.34	3.83	0.48	23.75	1.78	74,26
100/3/3	0.37	0.39	2.03	0.41	14.95	3.08	146,02
100/3/6	0.36	0.38	2.83	0.44	20.89	2.65	155,65
100/3/12	0.37	0.39	3.04	0.52	38.02	2.84	180,85
200/3/3	0.52	0.59	5.56	0.64	32.39	7.58	397,24
200/3/6	0.51	0.61	7.50	0.76	49.18	6.98	364,15
200/3/12	0.52	0.58	8.81	0.86	81.86	6.06	458,97
200/6/6	0.52	0.63	14.54	0.92	97.66	15.41	645,61
200/6/12	0.53	0.61	22.19	1.47	120.09	13.10	577,60
200/6/24	0.53	0.63	27.08	2.42	204.67	9.51	878,94
500/6/6	0.90	1.36	22.22	1.98	257.31	33.87	1630,22
500/6/12	0.88	1.32	35.43	2.64	330.59	29.60	1526,54
500/6/24	0.91	1.35	41.93	3.37	476.70	18.71	1104,32
500/12/12	0.88	1.41	109.52	4.69	674.44	78.05	1321,52
500/12/24	0.93	1.43	172.47	11.28	757.72	54.59	1864,04
500/12/48	0.93	1.57	213.77	18.68	1448.25	38.10	1678,38
1000/6/6	1.38	3.15	43.24	4.13	500.74	70.37	3600,00
1000/6/12	1.38	2.91	57.96	6.06	609.56	63.27	3600,00
1000/6/24	1.40	2.89	72.64	6.69	922.56	54.09	3600,00
1000/12/12	1.39	3.51	137.97	9.95	1574.51	163.34	2791,80
1000/12/24	1.38	3.52	219.97	20.08	1740.70	121.71	2882,86
1000/12/48	1.40	3.38	284.53	28.67	3129.72	88.82	3600,00

5.2 Problem instances with bidirectional lanes

In most of the papers on RMFS [5, 10, 29] the assumption is made that no collisions or deadlocks occur because unidirectional lanes are used in the storage area. However, with unidirectional lanes, some optimization potential is lost, since by allowing mobile robots to travel in the aisles in both directions the distance travelled can be reduced. The downside is that collisions and deadlocks will be significantly more likely to occur. Therefore, the collision avoidance strategy, explained in Section 4.3, is used to overcome this downside. In Figure 8, the simulation results are shown for the subset with three picking stations, six mobile robots and varying orders. Moreover, the MAS_{regret} is used, because it achieves good results very fast, in a warehouse setup with bidirectional lanes. However, the collision avoidance algorithm does slightly increase the computational time of the framework in comparison to the unidirectional scenario where no collision avoidance is required. This impact on the computational time increases with more AMRs in the system since the occurrences of conflicts grows, hence, the algorithm has to perform more conflict resolutions. For instance for operations with 3, 6, 12, 24, and 48 AMRs, on average 3%, 10%, 19%, 35%, and 61% of the routes that were performed in an instance had conflicts that had to be resolved. The MAS_{regret} with bidirectional lanes even performs better than the centralized scheduling algorithm of Teck & Dewil [19] in unidirectional lanes. This shows the significance of the optimization potential of bidirectional lanes. Important to note here is that the centralized algorithm does not include a collision avoidance strategy in its evaluation since this would increase its computational complexity even more. To fairly compare the results of the memetic algorithm with the MAS_{regret}, the generated schedules from the central algorithm in the bidirectional case are simulated in an environment with the collision avoidance mechanism.

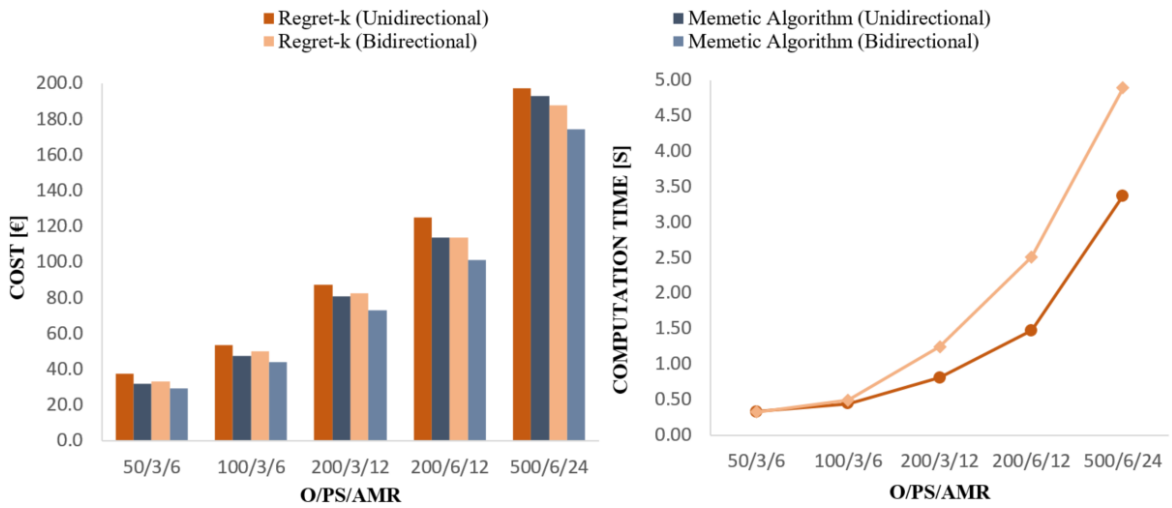

Figure 8. *Simulation results to compare unidirectional lanes scenario to the bidirectional lanes scenario.*

From the experiments in Figure 8, we see that the regret-k heuristic with bidirectional lanes is able to get a better performance than the unidirectional case. The performance is often even better than the performance of the centralized scheduler in the unidirectional case since the gain of the travelling in both directions on a lane allows for more performance gain than the gap in performance of the MAS_{regret} and the central algorithm. However, the central framework with bidirectional lanes does still outperform the MAS_{regret} with an average gap of 6,9%. To conclude, incorporating bidirectional lanes in the storage area of a RMFS allows for considerable optimization opportunities. This is to be expected since the AMRs can use the lanes in two directions, resulting in shorter possible routes. For instance, on average the routes

that the AMRs take on bidirectional lanes are 8,2 % shorter than those with unidirectional lanes. However, this positive impact on the system performance is partially undone by the occurrence of conflicts. This is especially the case for larger numbers of AMRs in the system. Still the benefit of shorter routes outweighs the additional waiting time of the AMRs and the rerouting since overall the MAS_{regret} in a bidirectional environments outperforms the MAS_{regret} in a unidirectional environment with on average 5,8% lower operational costs. Important to note, bidirectional lanes are only allowed if an effective collision and deadlock avoidance mechanism is in place.

5.3 Impact of the warehouse layout on system performance

All of the previous experiments were performed on a warehouse layout with the same number of inventory shelves as described in Table 2. To analyze the impact of different warehouse layouts on the system performance, both the number of inventory shelves and the layout of the shelves are varied to decrease (increase) the number of aisles and increase (decrease) the number of cross-aisles. The MAS_{regret} is used as the decision rule for the framework on an instance with 200 orders, 3 picking stations and 12 AMRs. Table 5 shows that, as expected, the warehouse layout has an impact on the system performance and in the strategic and tactical decision making this aspect has to be optimized. As for the operational decision making level, in a warehouse with more cross-aisles than aisles the AMRs have to travel considerably more to retrieve the inventory pods, which results in more costs. Moreover, the traffic density through the aisles, to travel to the picking stations, increases considerably since there are fewer options. This has a direct result on the number of conflicts that occur on bidirectional lanes, which is shown in the percentage of conflicts that occur on the total amount of routes travelled. In turn reduces the benefit of these bidirectional lanes since there are more waiting times and rerouting actions as shown by the gap between the costs of the unidirectional and bidirectional scenarios.

Table 5. *Impact of various warehouse layouts with 200 orders, 3 picking stations, and 12 AMRs.*

			Unidirectional		Bidirectional			
Shelves	Aisles	Cross-aisles	Cost [€]	Computational time [s]	Cost [€]	Computational time [s]	Conflicts [%]	Gap [%]
1200	23	4	82,1	1,25	74,7	1,63	21,3	9,0
1200	14	7	84,6	1,11	77,8	1,52	23,2	8,1
1200	5	19	107,6	1,13	103,3	2,23	40,2	4,0
1800	35	4	84,2	1,21	76,5	1,84	14,9	9,1
1800	17	9	87,4	0,86	82,6	1,24	21,2	5,5
1800	5	29	115,5	1,30	110,4	2,57	37,7	4,4

6 Conclusion and outlook

In this paper, we presented a multi-agent system for the simultaneous scheduling of AMRs and picking stations in an automated warehouse and we developed several different allocation mechanisms. The performance of the proposed system is compared to a centralized scheduling algorithm. From the experimental results we conclude that the $MAS_{LKH\text{-}3}$ is the best performing task allocation mechanism compared to the other developed mechanisms. Moreover, it is able to get good quality solutions compared to the solutions from the central algorithm in only of a fraction of the computation required by the central algorithm. Simple dispatching rules like the FIFO rule performed badly and the other task allocation mechanisms were not as competitive performance-wise compared to the $MAS_{LKH\text{-}3}$. However, the MAS_{regret} also obtains good quality solutions with a lot less computational effort than the centralized algorithm and

MAS_{LKH-3}. Moreover, it outperforms a dispatching method [40] that is used in a real-world RMFS application. Therefore, for large-scale instances where decisions have to be made quickly the MAS_{regret} rule is more favorable since it scales better with increasing retrieval tasks. Therefore, it is better suited for use in real-world optimization settings. Alternatively, in a situation where some time is available for optimization, the MAS_{LKH-3} is preferred since it reaches better quality solutions. For instance, the time between the release of a schedule for 100 picking orders and the actual execution of these orders, the MAS_{LKH-3} has enough time to generate a new schedule. In the special case where there is no time limit the centralized algorithm is still preferred. In conclusion, the proposed decentralized scheduling approach scales well with the number of picking tasks in the system and generates good quality solutions in a fraction of the time required by the centralized scheduling algorithm.

We also compare the proposed MAS_{regret} in a warehouse system with unidirectional lanes to a storage area with bidirectional lanes. From these experiments we see that the bidirectional lanes allow for a system performance that is better than the performance of the schedules obtained from the central algorithm with unidirectional lanes. However, bidirectional lanes can only be allowed if a collision avoidance strategy is in place since the assumption that no collisions or deadlocks occur is no longer valid.

To improve the ability of the multi-agent system to generate schedules close to the centrally generated schedules, future research can focus on hybridizing the two approaches where one can use a centralized scheduler to generate near-optimal schedules that can act as a blueprint for the agents in the multi-agent system to improve their decision making. Regarding the MAS_{LKH-3}, its computational complexity could be improved some more by making the re-optimization of the routes less nervous to changes in the environment. However, the expectation is that it would still be significantly slower than the MAS_{regret}. In this paper we assumed that picking times are deterministic. However, in reality this picking time is stochastic and may cause unforeseen additional waiting times for the AMRs waiting in the picking station queue, thus, impacting the complete schedule. The MAS should be able to dynamically adapt to these unforeseen delays and act accordingly. In this paper the velocity of the mobile robots is assumed to be constant, however, in future research it can be interesting to incorporate acceleration and deceleration capabilities for the AMRs. Speed optimization can make the avoidance of collisions and deadlocks more efficient since it allows for the mobile robots to continuously drive without having to wait. Thus, preserving its momentum.

7 Disclosure statement

No potential conflict of interest was reported by the authors.

8 Acknowledgments

This work was supported by the Research Foundation – Flanders (FWO) under Grant 1S97322.

9 References

[1] T. Gruchmann, A. Mies, T. Neukirchen, S. Gold, Tensions in Sustainable Warehousing: Including the Blue-Collar Perspective on Automation and Ergonomic workplace design. *J. Bus. Econ.* 91(2) (2020) 151-178. https://doi.org/10.1007/s11573-020-00991-1

[2] D. Truxillo, D. Cadiz, L. Hammer, Supporting the Aging Workforce: A Review and Recommendations for Workplace Intervention Research. *Ann. Rev. Org. Psych. Org. Beh.* 2(1) (2015) 351-381. https://doi.org/10.1146/annurev-orgpsych-032414-111435

[3] T. Le-anh, M.B.M. de Koster, A Review of Design and Control of Automated Guided Vehicle Systems, *Eur. J. Oper. Res.* 171 (2016) 1-23. https://doi.org/10.1016/j.ejor.2005.01.036
[4] L. Luo, N. Zhao, An Efficient Simulation Model for Layout and Mode Performance Evaluation of Robotic Mobile Fulfillment Systems. *Exp. Sys. Appl.* (2022) 117492. https://doi.org/10.1016/j.eswa.2022.117492
[5] M. Merschformann, T. Lamballais, M.B.M. de Koster, L. Suhl, Decision Rules for Robotic Mobile Fulfillment Systems. *Oper. Res. Persp.* 6 (2019) 100128. https://doi.org/10.1016/j.orp.2019.100128
[6] R. Van Lon, T. Holvoet, When Do Agents Outperform Centralized Algorithms?: A Systematic Empirical Evaluation in Logistics. *Aut. Agents Multi-Agent Sys.* 31(6) (2017) 1578-1609. https://doi.org/10.1007/s10458-017-9371-y
[7] R. Erol, C. Sahin, A. Baykasoglu, V. Kaplanoglu, A Multi-Agent Based Approach to Dynamic Scheduling of Machines and Automated Guided Vehicles in Manufacturing Systems. *Appl. S. Comp.* 12(6) (2012) 1720-1732. https://doi.org/10.1016/j.asoc.2012.02.001
[8] K. Chen, C. Chen, Applying Multi-agent Technique in Multi-section Flexible Manufacturing System. *Exp. Sys. Appl.* 37(11) (2012) 7310-7318. https://doi.org/10.1016/j.eswa.2010.04.024
[9] E. Macron, S. Chaabane, Y. Sallez, T. Bonte, A Multi-Agent System Based on Reactive Decision Rules for Solving the Caregiver Routing Problem in Home Health Care. *Sim. Model. Practice and Theory* 74 (2017) 134-151. https://doi.org/10.1016/j.simpat.2017.03.006
[10] Y. Bozer, F. Aldarondo, A Simulation-based Comparison of Two Goods-to-Person Order Picking Systems in an Online Retail Setting. *Int. J. Prod. Res.* 56(11) (2018) 3838-3858. https://doi.org/10.1080/00207543.2018.1424364
[11] C. Sahin, M. Demirtas, R. Erol, A. Baykasoglu, V. Kaplanoglu, A Multi-Agent Based Approach to Dynamic Scheduling with Flexible Processing Capabilities. *J. Int. Manuf.* 28 (2017) 1827-1845. https://doi.org/10.1007/s10845-015-1069-x
[12] S. Giordani, M. Lujak, F. Martinelli, A Distributed Multi-Agent Production Planning and Scheduling Framework for Mobile Robots. *Comp. Ind. Eng.* 64 (2013) 19-30. https://doi.org/10.1016/j.cie.2012.09.004
[13] M. De Ryck, D. Pissoort, T. Holvoet, E. Demeester, Decentral Task Allocation for Industrial AGV-systems with Resource Constraints. *J. Manuf. Sys.* 59 (2021) 310-319. https://doi.org/10.1016/j.jmsy.2021.03.008
[14] N. Boysen, D. Briskorn, S. Emde, Parts-to-Picker Based Order Processing in a Rack-moving Mobile Robots Environment. *Eur. J. Oper. Res.* 262(2) (2017) 550-562. https://doi.org/10.1016/j.ejor.2017.03.053
[15] X. Li, X. Yang, C. Zhang, M. Qi, A Simulation Study on the Robotic Mobile Fulfillment System in High-Density Storage Warehouses. *Sim. Model. Practice and Theory*, 122 (2021) 102366. https://doi.org/10.1016/j.simpat.2021.102366
[16] Zhuang, Y., Zhou, Y., Yuan, Y., Hu, X., & Hassini, E., 2022. Order Picking Optimization With Rack-Moving Mobile Robots and Multiple Workstations. *European Journal of Operational Research* 300: 527-544. https://doi.org/10.1016/j.ejor.2021.08.003
[17] X. Yang, G. Hua, L. Hu, T. Cheng, A. Huang, Joint Optimization of Order Sequencing and Rack Scheduling in the Robotic Mobile Fulfilment System. *Comp. Oper. Res.* 135 (2021) 105467. https://doi.org/10.1016/j.cor.2021.105467
[18] A. Valle, J. Beasley, Order Allocation, Rack Allocation and Rack Sequencing for Pickers in a Mobile Rack Environment. *Comp. Oper. Res.* 125 (2021) 105090. https://doi.org/10.1016/j.cor.2020.105090
[19] S. Teck, R. Dewil, A Bi-level Memetic Algorithm for the Integrated Order and Vehicle Scheduling in a RMFS. *Appl. S. Comp.* 121 (2022a) 108770. https://doi.org/10.1016/j.asoc.2022.108770
[20] S. Teck, R. Dewil, Optimization Models for Scheduling Operations in Robotic Mobile Fulfillment Systems. *Appl. Math. Model.* 111 (2022b) 270-287. https://doi.org/10.1016/j.apm.2022.06.036

[21] L. Xie, N. Thieme, R. Krenzler, H. Li, Introducing Split Orders and Optimizing Operational Policies in Robotic Mobile Fulfillment Systems. *Eur. J. Oper. Res.* 288 (2021) 80-97. https://doi.org/10.1016/j.ejor.2020.05.032

[22] A. Khamis, A. Hussein, A. Elmogy, A Multi-robot Task Allocation: a Review of the State-of-the-art. *Coop. Rob. Sens. Netw.* 8 (2015) 30-51.

[23] M. De Ryck, M. Versteyhe, F. Debrouwere, Automated Guided Vehicle Systems, State-of-the-art Control Algorithms and Techniques. *J. Manuf. Sys.* 54 (2020) 152-173. https://doi.org/10.1016/j.jmsy.2019.12.002

[24] B. Zou, Y. Gong, X. Xu, Z. Yuan, Assignment Rules in Robotic Mobile Fulfilment Systems for Online Retailers. *Int. J. Prod. Res.* 55 (2017) 6175-6192. https://doi.org/10.1080/00207543.2017.1331050

[25] A. Gharehgozli, N. Zaerpour, Robot Scheduling for Pod Retrieval in a Robotic Mobile Fulfillment System. *Trans. Res. Part E* (2020)142. https://doi.org/10.1016/j.tre.2020.102087

[26] S. Satunin, E. Babkin, A Multi-Agent Approach to Intelligent Transportation Systems Modelling with Combinatorial Auctions. *Exp. Sys. Appl.* 41(15) (2014) 6622-6633. https://doi.org/10.1016/j.eswa.2014.05.015

[27] Y. Liu, S. Sun, X. Wang, L. Wang, An Iterative Combinatorial Auction Mechanism for Multi-agent Parallel Machine Scheduling. *Int. J. Prod. Res.* 60(1) (2021) 361-380. https://doi.org/10.1080/00207543.2021.1950938

[28] F. Weidinger, N. Boysen, Scattered storage: How to Distribute Stock Keeping Units All Around a Mixed-Shelves Warehouse. *Trans. Sc.* 52(6) (2018) 1412-1427. https://doi.org/10.1287/trsc.2017.0779

[29] T. Lamballais, D. Roy, M. de Koster, Estimating Performance in a Robotic Mobile Fulfillment Sytem. *Eur. J. Oper. Res.* 256(3) (2017) 976-990. https://doi.org/10.1016/j.ejor.2016.06.063

[30] J. Enright, P.R. Wurman, Optimization and Coordinated Autonomy in Mobile Fulfillment Systems. *In Proceedings of the AAAI workshop on automated action planning for autonomous mobile robots* (2011) 33-38.

[31] R. Meyer, Event-Driven Multi-Agent Simulation. *International Workshop on Multi-Agent Systems and Agent-Based Simulation* Spring, (2014) 3-16.

[32] B. Jin, W. Zhu, A. Lim, Solving The Container Relocation Problem by an Improved Greedy Look-ahead Heuristic. *Eur. J. Oper. Res.* 240(3) (2014) 837-847. https://doi.org/10.1016/j.ejor.2014.07.038

[33] J. Potvin, J. Rousseau, A Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows. *Eur. J. Oper. Res.* 66(3) (1993) 331-340. https://doi.org/10.1016/0377-2217(93)90221-8

[34] Z. Chen, J. Alonso-Mora, X. Bai, D. Harabor, P. Stuckey, Integrated Task Assignment and Path Planning for Capacitated Multi-Agent Pickup and Delivery. *IEEE Robotics and Automation Letters* 6(3) (2021) 5816-5823. https://doi.org/10.1109/LRA.2021.3074883

[35] V. Hemmelmayr, J. Cordeau, T. Crainic, An Adaptive Large Neighborhood Search Heuristic for Two-Echelon Vehicle Routing Problems Arising in City Logistics. *Comp. Oper. Res.* 39(12) (2012) 3215-3228. https://doi.org/10.1016/j.cor.2012.04.007

[36] K. Helsgaun, An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems. *Technical Report* (2017).

[37] S. Lin, W. Kernighan, An Effective Heuristic Algorithm for the Travelling-Salesman Problem. *Oper. Res.* 21 (1973) 498-516. https://doi.org/10.1287/opre.21.2.498

[38] Z. Zhang, Q. Guo, J. Chen, P. Yuan, Collision-Free Route Planning for Multiple AGVs in an Automated Warehouse Based on Collision Classification. *IEEE Access: Special selection on human-centered smart systems and technologies* 6 (2018) 26022-26035. https://doi.org/10.1109/ACCESS.2018.2819199

[39] M. Zhong, Y. Yang, Y. Dessouky, O. Postolache, Multi-AGV Scheduling for Conflict-Free Path Planning in Automated Container Terminals. *Comp. Ind. Eng.* 142 (2020) 106371. https://doi.org/10.1016/j.cie.2020.106371

[40] H. Qin, J. Xiao, D. Ge, L. Xin, J. Gao, S. He, H. Hu, JD.com: Operations Research Algorithms Drive Intelligent Warehouse Robots to Work. *INFORMS Journal on Applied Analytics* 52(1) (2022) 42-55. https://doi.org/10.1287/inte.2021.1100

10 Appendices

Appendix A. Simulation results MAS framework with FIFO dispatching rule.

O/PS/AMR	Time [s]	Makespan [s]	Total distance [m]	Server utilization	FIFO [€]
50/3/3	0.33	3296.9	10416	11.5%	69.7
50/3/6	0.35	1716.4	10778	22.0%	45.7
50/3/12	0.34	919.3	11155	40.6%	33.9
100/3/3	0.37	4983.9	15763	12.1%	105.5
100/3/6	0.36	2598.4	16206	23.0%	69.0
100/3/12	0.37	1295.8	15907	46.0%	48.1
200/3/3	0.52	11785.8	37239	12.0%	249.3
200/3/6	0.51	5939.47	37274	23.7%	158.0
200/3/12	0.52	3069.2	37674	44.8%	113.9
200/6/6	0.52	5940.9	37648	11.8%	251.5
200/6/12	0.53	3058.1	38446	22.8%	162.9
200/6/24	0.53	1596.7	38626	42.3%	117.5
500/6/6	0.90	13612.6	86043	12.1%	576.0
500/6/12	0.88	6475.6	84705	25.4%	350.6
500/6/24	0.91	3490.9	85872	46.1%	259.4
500/12/12	0.88	6328.9	83039	13.0%	540.9
500/12/24	0.93	3224.4	83736	25.3%	348.1
500/12/48	0.93	1735.3	85697	46.4%	258.4
1000/6/6	1.38	27106.5	171711	12.0%	1147.6
1000/6/12	1.38	13684.7	172275	23.7%	729.1
1000/6/24	1.40	7091.7	173747	45.9%	525.7
1000/12/12	1.39	13258.2	167377	12.3%	1121.6
1000/12/24	1.38	6699.8	167352	24.3%	711.6
1000/12/48	1.40	3433.6	167485	46.6%	507.7

Appendix B. Simulation results MAS framework with SSIA mechanism.

O/PS/AMR	Time [s]	Makespan [s]	Total distance [m]	Server utilization	SSIA [€]
50/3/3	0.32	2974.5	9384	12.8%	62.9
50/3/6	0.33	1487.3	9202	25.0%	39.3
50/3/12	0.34	862.7	9582	42.0%	30.3
100/3/3	0.39	4687.6	14622	12.8%	98.8
100/3/6	0.38	2265.9	13838	26.4%	59.6
100/3/12	0.39	1195.6	13150	50.5%	41.7
200/3/3	0.59	9689.5	29650	14.6%	203.3
200/3/6	0.61	4774.9	28828	29.3%	125.1
200/3/12	0.58	3631.6	28729	38.0%	107.0
200/6/6	0.63	4939.0	30419	14.2%	207.6
200/6/12	0.61	2487.6	30040	27.8%	130.3
200/6/24	0.63	1497.6	28340	45.6%	96.4
500/6/6	1.36	9950.7	59436	16.6%	415.0
500/6/12	1.32	6093.3	64613	26.6%	303.5
500/6/24	1.35	3813.8	62139	40.3%	227.9

500/12/12	1.41	5786.2	64177	14.3%	474.0
500/12/24	1.43	2995.8	64564	27.1%	300.8
500/12/48	1.57	1806.7	63353	42.3%	223.8
1000/6/6	3.15	18859.2	111776	17.3%	785.0
1000/6/12	2.91	16493.3	127693	19.6%	738.9
1000/6/24	2.89	9067.1	119114	35.6%	491.8
1000/12/12	3.51	11563.6	121715	14.0%	935.7
1000/12/24	3.52	6569.1	114132	24.1%	610.3
1000/12/48	3.38	4072.7	115554	37.7%	456.8

Appendix C. Simulation results MAS framework with Look-Ahead heuristic.

O/PS/AMR	Time [s]	Makespan [s]	Total distance [m]	Server utilization	Look-Ahead [€]
50/3/3	1.90	2752.9	8356	13.7%	57.6
50/3/6	2.72	1441.0	8655	25.9%	37.7
50/3/12	3.83	786.5	8683	45.0%	27.5
100/3/3	2.03	4352.1	13096	13.8%	90.9
100/3/6	2.83	2197.5	12713	27.3%	56.6
100/3/12	3.04	1221.8	12917	48.0%	41.7
200/3/3	5.56	9868.2	29762	14.2%	206.3
200/3/6	7.50	5101.9	29236	27.4%	130.9
200/3/12	8.81	3030.5	29099	46.2%	98.3
200/6/6	14.54	5111.2	28473	13.9%	209.6
200/6/12	22.19	2703.9	29141	25.7%	135.5
200/6/24	27.08	1429.8	28234	46.3%	94.1
500/6/6	22.22	11401.2	64818	14.6%	469.7
500/6/12	35.43	5614.1	65239	29.1%	289.6
500/6/24	41.93	3719.5	62007	44.2%	224.8
500/12/12	109.52	5659.8	66961	14.6%	470.9
500/12/24	172.47	3125.2	66924	27.0%	312.4
500/12/48	213.77	1786.2	64002	45.3%	223.6
1000/6/6	43.24	24233.7	128957	13.5%	983.0
1000/6/12	57.96	13026.4	118231	25.0%	614.0
1000/6/24	72.64	8539.9	120629	36.9%	478.0
1000/12/12	137.97	11978.4	128001	13.6%	972.7
1000/12/24	219.97	6064.1	121884	26.5%	592.3
1000/12/48	284.53	3872.3	118864	40.8%	450.0

Appendix D. Simulation results MAS framework with the dispatching method of Qin et al. (2022).

O/PS/AMR	Time [s]	Makespan [s]	Total distance [m]	Server utilization	Qin et al. [€]
50/3/3	2.20	2749.6	8152	13.9%	57.2
50/3/6	1.91	1405.6	8210	27.3%	36.3
50/3/12	1.78	777.2	8674	49.3%	27.3
100/3/3	3.08	4134.5	12062	14.7%	85.7
100/3/6	2.65	2115.8	12211	28.7%	54.4
100/3/12	2.84	1278.5	13753	47.5%	44.0
200/3/3	7.58	9643.9	28294	14.7%	200.2
200/3/6	6.98	5086.2	29670	27.8%	131.4
200/3/12	6.06	2776.1	31416	50.9%	98.4
200/6/6	15.41	4666.9	27186	15.1%	193.4
200/6/12	13.10	2621.6	29538	27.0%	133.6
200/6/24	9.51	1444.7	30831	48.9%	99.1
500/6/6	33.87	11444.6	66581	14.5%	474.2

500/6/12	29.60	5751.9	68147	28.8%	299.0
500/6/24	18.71	3287.5	70947	50.4%	226.9
500/12/12	78.05	5274.7	63650	15.7%	441.1
500/12/24	54.59	2836.4	65968	29.2%	292.7
500/12/48	38.10	1600.3	69700	51.8%	222.0
1000/6/6	70.37	22099.5	128736	14.8%	915.9
1000/6/12	63.27	11753.8	131213	27.8%	596.9
1000/6/24	54.09	6321.1	139684	51.7%	442.0
1000/12/12	163.34	11556.9	131600	14.2%	952.6
1000/12/24	121.71	5961.9	132449	27.4%	604.4
1000/12/48	88.82	3158.2	135700	51.8%	434.9

Appendix E. Simulation results MAS framework with regret-k heuristic.

O/PS/AMR	Time [s]	Makespan [s]	Total distance [m]	Server utilization	Regret-k [€]
50/3/3	0.32	2974.5	9384	12.8%	62.9
50/3/6	0.33	1487.3	9202	25.0%	39.3
50/3/12	0.34	862.7	9582	42.0%	30.3
100/3/3	0.39	4687.6	14622	12.8%	98.8
100/3/6	0.38	2265.9	13838	26.4%	59.6
100/3/12	0.39	1195.6	13150	50.5%	41.7
200/3/3	0.59	9689.5	29650	14.6%	203.3
200/3/6	0.61	4774.9	28828	29.3%	125.1
200/3/12	0.58	3631.6	28729	38.0%	107.0
200/6/6	0.63	4939.0	30419	14.2%	207.6
200/6/12	0.61	2487.6	30040	27.8%	130.3
200/6/24	0.63	1497.6	28340	45.6%	96.4
500/6/6	1.36	9950.7	59436	16.6%	415.0
500/6/12	1.32	6093.3	64613	26.6%	303.5
500/6/24	1.35	3813.8	62139	40.3%	227.9
500/12/12	1.41	5786.2	64177	14.3%	474.0
500/12/24	1.43	2995.8	64564	27.1%	300.8
500/12/48	1.57	1806.7	63353	42.3%	223.8
1000/6/6	3.15	18859.2	111776	17.3%	785.0
1000/6/12	2.91	16493.3	127693	19.6%	738.9
1000/6/24	2.89	9067.1	119114	35.6%	491.8
1000/12/12	3.51	11563.6	121715	14.0%	935.7
1000/12/24	3.52	6569.1	114132	24.1%	610.3
1000/12/48	3.38	4072.7	115554	37.7%	456.8

Appendix F. Simulation results MAS framework with LKH-3 heuristic.

O/PS/AMR	Time [s]	Makespan [s]	Total distance [m]	Server utilization	LKH [€]
50/3/3	9.79	2581.2	7633	14.5%	53.7
50/3/6	14.17	1363.5	7606	27.2%	34.6
50/3/12	23.75	835.7	7941	42.2%	27.0
100/3/3	14.95	4160.4	12250	14.6%	86.4
100/3/6	20.89	2068.3	11588	28.2%	52.6
100/3/12	38.02	1188.2	11925	50.9%	39.4
200/3/3	32.39	8797.3	25687	15.9%	182.4
200/3/6	49.18	4625.2	26210	29.9%	118.1
200/3/12	81.86	2694.5	26367	52.2%	88.2
200/6/6	97.66	4444.1	25650	15.8%	183.8
200/6/12	120.09	2350.8	25288	29.5%	117.7
200/6/24	204.67	1358.3	26086	50.6%	88.1

500/6/6	257.31	10073.6	57999	16.4%	416.3
500/6/12	330.59	5066.3	58639	32.3%	260.9
500/6/24	476.70	2861.4	54885	54.6%	185.5
500/12/12	674.44	4775.9	56462	17.1%	397.3
500/12/24	757.72	2585.8	57284	31.1%	261.9
500/12/48	1448.25	1551.6	56959	52.7%	196.7
1000/6/6	500.74	19675.2	112923	16.6%	812.5
1000/6/12	609.56	9840.6	107877	33.1%	496.3
1000/6/24	922.56	5622.5	107533	57.4%	363.9
1000/12/12	1574.51	10017.3	115646	16.2%	828.5
1000/12/24	1740.70	4946.6	107080	32.9%	496.6
1000/12/48	3129.72	2780.9	104701	56.6%	357.0

Appendix G. Simulation results memetic algorithm (Teck and Dewil, 2022a).

O/PS/AMR	Time [s]	Makespan [s]	Total distance [m]	Server utilization	MA [€]
50/3/3	56,04	2453.6	7295	15.4%	51.1
50/3/6	57,73	1237.6	7307	29.5%	32.1
50/3/12	74,26	750.7	7415	48.9%	24.7
100/3/3	146,02	3509.2	10222	17.1%	72.7
100/3/6	155,65	1839.5	10639	19.0%	47.4
100/3/12	180,85	963.3	10292	61.0%	33.1
200/3/3	397,24	8686.5	25669	16.2%	180.7
200/3/6	364,15	4384.2	25624	32.0%	113.4
200/3/12	458,97	2371.6	24969	59.1%	80.8
200/6/6	645,61	4218.7	24824	16.6%	175.3
200/6/12	577,60	2257.3	24696	30.7%	113.8
200/6/24	878,94	1230.9	24804	57.4%	80.2
500/6/6	1630,22	10203.6	60267	16.2%	424.3
500/6/12	1526,54	4985.3	59465	32.9%	259.9
500/6/24	1104,32	2923.5	58107	55.9%	193.1
500/12/12	1321,52	4786.2	57305	17.2%	399.4
500/12/24	1864,04	2779.9	54673	29.5%	269.4
500/12/48	1678,38	1735.6	53829	47.1%	202.7
1000/6/6	3600,00	20979.0	124552	15.6%	873.6
1000/6/12	3600,00	10419.4	118598	31.3%	533.2
1000/6/24	3600,00	5535.7	114528	58.7%	373.4
1000/12/12	2791,80	10311.1	119031	15.8%	852.8
1000/12/24	2882,86	5959.5	113107	27.3%	570.4
1000/12/48	3600,00	3424.7	108015	47.4%	403.1