Decentral task allocation for industrial AGV-systems with routing constraints

Non Peer-reviewed author version

De Ryck, M.; Pissoort, D.; Holvoet, T. & DEMEESTER, Eric (2022) Decentral task allocation for industrial AGV-systems with routing constraints. In: JOURNAL OF MANUFACTURING SYSTEMS, 62 , p. 135 -144.

DOI: 10.1016/j.jmsy.2021.11.012
Handle: http://hdl.handle.net/1942/37268

Decentral Task Allocation for Industrial AGV-Systems with Routing Constraints

M. De Ryck[a,*], D. Pissoort[a], T. Holvoet[b], E. Demeester[c]

[a] *Faculty of Engineering Technology, KU Leuven, Spoorwegstraat 12, 8200 Bruges, Belgium*
[b] *Faculty of Engineering Science, KU Leuven, Celestijnenlaan 200a, 3001 Leuven, Belgium*
[c] *Faculty of Engineering Technology, KU Leuven, Agoralaan B, 3590 Diepenbeek, Belgium*

Abstract

Automated Guided Vehicles (AGVs) form a large and important part of the logistics transportation systems in today's industry and are widely used. One of the main problems in the control of AGV systems is the routing problem, where all robots attempt to execute all of their allocated tasks without congestions with other robots. This is a complex problem and a large variety of solutions exist in literature to this purpose. One possible solution that is not widely investigated in literature is the possibility of including routing information as a constraint in the task allocation process in order to obtain a more balanced allocation that could reduce possible congestions. In task allocation literature, costs for a robot to execute a task are largely estimated without considering possible delays that can occur due to congestions when moving to a task. In this paper, the observed research gap of including routing constraints in the task allocation problem is addressed by proposing a decentralized task allocation algorithm based on sequential single-item (SSI) auctions with the implementation of delegate-MAS (DMAS) as a routing constraint in the bidding process. Our new approach is benchmarked to an SSI-solver that does not take routing constraints into account.

Keywords: Automated Guided Vehicles, Decentralization, Task Allocation, Routing Constrained

1. Introduction

Automated Guided Vehicles (AGVs) are mobile robots that perform transportation tasks in all types of applications: from e-commerce warehouses over material handling in assembly lines, to pharmacy, and further. AGV systems are usually controlled in a centralized way where one single fleet manager coordinates the whole fleet. However, our research focuses largely on a decentralized control of AGV-systems as they can offer some benefits over centralized control [1–3]. In decentralized control of a mobile robot fleet, robots use only local information to make decisions and hence, participate in the optimization process. This decentral decision making does not necessarily mean that the robot algorithms need to be situated on the separate hardware devices of the robots, but might as well be situated in one piece of hardware. When the decentral decision making algorithms for task allocation are situated on separate hardware, then this is called distributed task allocation. But this paper proposes a decentral algorithm. In [4], we made a review on control algorithms and techniques in AGV systems with a large focus on decentralized control. More specifically, in our research, we investigate decentralized task allocation under constraints. In [5], we targeted the task allocation problem considering resource

*Corresponding author
Email addresses: matthias.deryck@kuleuven.be (M. De Ryck), davy.pissoort@kuleuven.be (D. Pissoort), tom.holvoet@kuleuven.be (T. Holvoet), eric.demeester@kuleuven.be (E. Demeester)

constraints by introducing an intelligent resource management approach [6] that showed good results. This paper will target the task allocation problem considering routing constraints in order to account for possible delays when moving towards a task due to roadblocks or other AGVs moving. This is done by implementing routing information in the bidding mechanism of a sequential single-item (SSI) auction process, making the routing model used in the bidding process more realistic. In general, path costs for executing a new task in a task allocation process are usually computed using, for example, an A*-shortest path planner that does not consider possible congestions or routing delays. It is assumed that the robot can execute its task and that no roadblocks occur or that no road segments are occupied by other robots. In our approach, routing information is considered in the bidding process. AGVs bid on tasks considering the possibility of a detour if some road segments are blocked dynamically or if they are already occupied by other robots at the required time slot. It is assumed that robots can allocate road segments at particular slots in time to prevent multiple robots of allocating one segment at a time. This resource allocation results in a task allocation approach with a more realistic model of the AGV routing. For resource allocation and for obtaining routing information, the delegate-MAS routing approach [7–9] is used. Fusion of routing in the task allocation process requires few adaptions to the control architecture, facilitating their implementation in industrial AGV-systems. Beside AGV-systems, this approach could be of benefit to all systems where a fleet of vehicles needs to cooperate in order to distribute a set of transportation tasks among each other in highly dynamic environments, e.g. parcel delivery, material handling, etc. In any of these systems, it is of primordial importance to reduce the possibility of congestions and to be able to estimate task execution times as accurately as possible. These are two properties that result from our approach and could therefore be of relevance to multiple industrial systems.

The paper is organized as follows: Section 2 covers the literature on routing. Section 3 covers the proposed decentral task allocation architecture under routing constraints. Section 4 illustrates the proposed approach with a concrete example. Section 5 validates the proposed architecture in simulation and discusses the results. Section 6 draws concluding remarks.

2. Literature review

Besides the task scheduling problem in multi-robot systems, the multi-robot routing problem is a vast research area that gained much attention in the past decades. Multi-robot task scheduling deals with allocating the tasks to robots in the system. We reviewed the multi-robot task allocation literature in our previous work [4, 5]. Multi-robot routing, on the other hand, deals with finding the most optimal path for each robot in the system in order to execute all assigned tasks resulting from the task scheduling problem, and this without congestions, deadlocks, or live-locks. This is a complex problem, especially when robots are restricted to move on guide-path networks that can consist of both bi- or unidirectional paths as these put large restrictions on the geographical freedom of the robots. Over the years, many solutions to this problem have been proposed. One approach to avoid congestions is to engineer the guide-path network in such a way that congestions are avoided as much as possible. However, the effort to engineer such networks is significant and limits the freedom of customers to have the network layout as they desire for their application.

One effective method to avoid congestions by means of control algorithms is called zone control [10, 11]. Zone control divides the total guide-path network in distinct zones where only a limited number of robots is allowed. This is highly effective as this decreases the chance of having a congestion significantly. However, zone control results in highly sub-optimal behavior as not all robots

can have access to the whole facility at the same moment, which will induce waiting times for entering particular zones.

In addition to zone control, many routing algorithms exist in order to provide each robot with optimal paths while avoiding congestions. In general, these algorithms can be divided into centralized and decentralized routing algorithms:

- Centralized routing algorithms [12–14] do the routing for all robots simultaneously, before execution, while having global information. For example, they can provide an optimal solution using a prioritized routing with A*, where robots are assigned non-conflicting routes in a consecutive order, depending on their priority, by planning the routes considering the already planned routes of higher prioritized robots. This type of routing is also known as static routing because it is scheduled before the tasks are executed. This can ensure congestion-free paths before runtime, but can easily fail if changes occur during execution. In addition, central approaches typically inherit some other drawbacks, such as increased computation with increasing number of robots in the system, and the need for a complete re-planning of the routing solution when dynamic changes occur, such as roadblocks, robot fallout's, etc.

- Decentralized routing algorithms [15–17] attempt to overcome these limitations. In these methods, robots plan their routes individually only using local information. In this way, rerouting due to roadblocks can be performed without much computational effort, resulting in high adaptation to changing environments. This is also known as dynamic routing because the congestion-free solution can be dynamically adjusted during runtime, making the routing more robust against changing circumstances. In addition, robustness can be increased because the routing of robots is not affected by the fallout of other robots. A disadvantage is that decentralized solutions usually provide sub-optimal solutions because they use only local information.

A simple way of decentral dynamic routing is rule-based routing [18] that defines a set of specific movement rules for robots when they proceed, this to reduce complexity. In many industrial AGV systems, rule-based dynamic routing is used to avoid congestions. The common routing algorithms used in the industry attempt to plan optimal and congestion-free routes without considering the availability of the resources they need, which are the physical roads and intersections of the guide-path network. When not considering these layout parts as resources that can be allocated or reserved, an imported dimension in which can be planned besides the spatial dimension is skipped, namely time. When nodes and intersections are modeled as resources that can be allocated to robots in time, much more complex routing could be managed. To this end, some routing algorithms exist that take into account resource allocation [7, 19]. A promising decentral and dynamic routing approach that uses resource allocation is called delegate-MAS [7–9]. Delegate-MAS (Delegate Multi-Agent Systems) makes use of a virtual representation of the environment in which segments, nodes, and intersections all maintain a reservation schedule in which robots can make reservations in time. The dynamic routing process acts in a belief-desire-intention (BDI) manner in which each robot independently (i) updates its believes about the environment in which it looks for a set of possible routes through the virtual guide path network, (ii) sets some desires in which it evaluates all routes minimizing a certain objective, and (iii) effectuates its desire by reserving the required path segments in time. It does this by using ant colonization principles (ACO) [20] in which a set of lightweight ants traverse the virtual guided path

network looking for possible routes and virtually execute each route, poll nodes for their schedule, and calculate the time a particular route would take. This BDI-process is repeated frequently to respond to changing circumstances. When a road gets dynamically blocked, the ants find alternative routes and update their intention by reserving the new route.

This delegate-MAS routing principle will be used to account for routing constraints in our proposed decentral task allocation approach presented in this paper. Research on the implementation of temporal and precedence constraints in decentral multi-robot task allocation has already been extensively investigated in the literature [21, 22], but the implementation of routing constraints has not yet been investigated, to the authors' knowledge, and is thus investigated in this paper.

3. Proposed task allocation architecture under routing constraints

Our proposed decentral task allocation approach uses an auction-based task allocation process and more specifically the sequential single-item (SSI) auction process [23]. This SSI-approach is used as this is proven to have the best trade-off between optimality and computational efficiency and considers robots as bidders, and tasks as goods [23–25]. In Section 3.1, the sequential single-item auction process and how it is implemented using a linear combination of two bid principles, MiniSum and MiniMax [26], is elaborated. In Section 3.2, our contribution to this approach, by implementing delegate-MAS as a routing constraint, is discussed. In Section 3.3, the total auction algorithm with routing constraints is elaborated.

3.1. Task allocation approach

3.1.1. Setup

A set of tasks $T = \{t_1, t_2, \ldots, t_m\}$ needs to be allocated to a set of robots $R = \{r_1, r_2, \ldots, r_n\}$ that can be used to execute the tasks. Tasks are defined as locations that robots need to visit. Robots can have multiple tasks they are assigned to, represented by a local task list LT^i of robot r_i, which is initially empty. Our proposed architecture is a decentralized architecture, this means that most of the knowledge is decentralized. To participate into the auctions, each robot only needs local information: its own location, local task list, a graph of the layout, and information on the newly announced task. The layout on which the robots move consist of depot stations $D = \{d_1, d_2, \ldots, d_n\}$ at which no tasks can be spawned, and other stations $S = \{s_1, s_2, \ldots, s_k\}$ at which tasks can be spawned. The total graph of the layout is presented as G with vertices or nodes $V = D \cup S$, and segments or edges E, which is a set of edges joining any two vertices from V.

3.1.2. Auction process

All tasks to execute T are initially unallocated and are announced all together for auction to all available robots R by an auctioneer. After receiving the announcement of a new task T_{new} from an auctioneer, all robots compute the cost for including this task in their local task list as a bid, following some bidding rule described further. After bid computation, all robots send their bids to the auctioneer which then determines the winning task, and winning robot of that task, following some winner determination rule described further. The winning robot finally adds the task to its ordered local task list in an optimal way and obtains a new local task list $LT'^i = LT^i \bigcap T_{new}$. To find the optimal order of the tasks in LT'^i, a Traveling Salesman Problem (TSP) [27] could be solved with any solution approach. After the allocation of T_{new}, the other tasks in T are sequentially auctioned in the same way until there are no tasks left in T.

Bidding phase. When a robot receives a task from an auctioneer, it computes a bid b that represents the cost of executing this task. Our approach uses a combination of the

MiniSum and the MiniMax bidding rules by calculating the MiniSum and MiniMax bids, b_{ms} and b_{mm}, for each robot $r_i \in R$, and combine them using a linear combination. The MiniSum bidding rule results in a minimized total path cost summed over all robots. The MiniMax bidding rule results in a minimum total time span for executing all tasks. Both of them could be beneficial in certain situations. For this reason, an extra parameter $\varepsilon \in [0, 1]$ was proposed that is tune-able by the user or which can be changed/learned automatically during robot execution. The total bid that is sent to the auctioneer is a linear combination of both bidding rules:

$$b = \varepsilon \cdot bms + (1 - \varepsilon) \cdot b_{mm} \tag{1}$$

When ε is 1, the user is focused on the total path cost and wants the robots to consume as little energy as possible. This can be the case when there is a moment of fewer transports and when the available time can be used to refill most of the robots and having few robots executing tasks. When ε is 0, the user is focused on the total execution time and wants this to be as low as possible. This can be the case during rush periods when many tasks need to be executed urgently. When the user wants both objectives to be minimized equally, ε can be put to 0.5.

- To calculate the MiniSum bid b_{ms}, robots need to estimate the extra cost to insert the new task T_{new} in their local task list LT^i. First, they estimate the total cost c_1 for executing their initial task list LT^i. Second, they estimate the total cost c_2 for executing a new local task list $LT'^i = LT^i \bigcap T_{new}$ where the optimal order is found via solving a TSP problem. Finally, they compute the difference of both costs to obtain the MiniSum bid.

$$b_{ms} = c_2 - c_1$$

- To calculate the MiniMax bid b_{mm}, robots only need to estimate the total cost c_2, as in this approach, robots only bid the total cost of executing the new task, so no marginal cost is calculated.

$$b_{mm} = c_2$$

When both bids are calculated, this is combined using Eq. 1 to obtain the final bid that is sent to the auctioneer for the winner determination phase. For the bid computation, costs are defined as times in seconds. The cost of moving from one location to another in the graph is calculated using the distance obtained from the A* shortest path planning algorithm [28] and the vehicle's velocity v.

Winner Determination phase. As a winner determination phase, the simple Lowest Bid (LB) principle [29] is adopted. The auctioneer receives all bids from the robots at each task and assigns the task that has the lowest bid to the robot that computed that bid. This principle is used because this gives us directly the minimization effect of our objective. Using this principle in combination with our bidding rule, we aim at minimizing the total cost of executing all tasks, as well as the maximum time span in which all tasks are executed.

3.2. Routing approach

When routing constraints are considered in this auction process, only minor adaptions need to be made to both the setup and the auction process of the classical SSI-auction approach. The routing constraints that will be included in the bidding process will come from the delegate-MAS routing approach. Using the implementation of delegate-MAS as routing constraint in the bidding mechanism, it is attempted to avoid congestions before they occur by maintaining a proper load balance in the environment layout and by reserving time slots at specific layout segments.

3.2.1. Setup

In order to consider routing constraints, robots must be able to verify that all locations in their local task list

LT_i can be visited without collisions or deadlocks. To this purpose, each robot needs access to a virtual representation of the layout, which is the graph G. To implement the delegate-MAS routing approach, every node V in the original graph G must be extended with a software piece that monitors the node's state (blocked, crowded, etc.) and that holds a reservation schedule, which is modeled as a Simple Temporal Network (STN) [30] where nodes represent points in time and edges represent time intervals. Such a software piece will be called an environmental agent. Each robot r_i can make reservations in the schedule of each environmental agent to reserve the node for a specific period of time in the future. Initially, all of the environmental agents' schedules are empty.

3.2.2. Auction process

In order to consider routing constraints, this only needs little adaption to the bid calculation from Section 3.1.2. This boils down to obtaining a route using the DMAS-algorithm towards all tasks in the sequence that needs to be evaluated. Thus for every task, all feasible paths towards the task are evaluated, and for the best path among them, the total cost including delays is calculated. The total cost for executing all tasks is further used in the bidding process. If the robot has a local task list LT_i' including a new task T_{new}, for which it needs to evaluate the cost of accepting the task, delegate-MAS calculates the total cost of executing these tasks including extra delays that result from congestion avoidance. The delegate-MAS process occurs in three phases:

1. **Feasibility phase:** For scheduling a route over all tasks in LT_i', a set of feasibility ants propagates through the virtual network G looking for a set of feasible paths that each contain at least all locations in the local task list LT_i'. The robot retrieves the feasible paths from the ants and uses them for the next phase.
2. **Exploration phase:** A set of exploration ants propagates through the virtual network G following each of the feasible paths obtained from the feasibility phase in order to estimate the total execution time of each of the paths including possible delays caused by (temporarily) blocked nodes. Starting from the next node the robot will visit, and at the time the robot starts at this node, the ants virtually execute the route by visiting all nodes in sequence while polling each node's environmental agent to see if the node is free at the time the ants arrive, considering the traveled time already executed. If the node is free at the time slot the ant needs to traverse the node, no extra delay is included in the total travel time. If the node is not free on the other hand, the environmental agent reports the shortest delay the ant needs to wait before the node is free. The ant considers this extra delay in its total travel time. When the ant reaches the end of the path, it knows how long this path would take for the robot to traverse including the delays caused by already reserved nodes of other robots. The ant keeps the list of time slots that it can get for each node in its route. The robot retrieves all total travel times and slots for all feasible paths from the ants and selects the best path following some objective, which is the shortest traveling time including delays in our approach. The cost for executing this route, including the new task T_{new}, can be used in the bidding process.
3. **Intention phase:** If the task is assigned to the robot by the auctioneer, the obtained path needs to be fixed in the environment. A set of intention ants propagates through the virtual network G following the chosen best path from the exploration phase. Starting from the next node the robot will visit, and at the time the robot starts at this node, the ants visit all nodes in the path in sequence and

make reservations at each node for the required time slot calculated in the exploration phase. A node is always reserved for a time slot (t_1, t_2) with t_1 the time of moving towards the node (thus leaving the previous node), and t_2 the time of arriving at the node. The ants make reservations at all nodes for the calculated time slots. This results in a fully reserved path in time, which can be used as a constraint in the route planning of other robots as only one robot can occupy a node at the same time.

Algorithm 1 shows the pseudo-code of the delegate-MAS process. The algorithm takes the starting point p, the set of locations to visit LT, and the graph of the layout G as an input.

Algorithm 1 DMAS-algorithm

```
1: procedure DMAS(p, LT, G)
2:     feasible_paths ← feasibility_phase(p, LT, G)
3:     path, slots ← exploration_phase(feasible_paths, G)
4:     return path, slots
```

The implementation of the routing constraints come into play at the calculation of the bidding costs c_1 and c_2. For c_1, the robot looks at its current locations it needs to visit and uses the DMAS-algorithm to obtain an optimal route (using LT_i), and receiving the total time cost for executing this route considering possible delays. For c_2, the robot considers the new task in its local task list and also uses the DMAS-algorithm to obtain the new route (using LT_i'). The total bid is again computed using Eq. 1. When the bidding process is finished and a robot is assigned the task, this robot performs an intention phase in which it reserves its new route into the environment. When including these extra routing constraints in the bidding process, delays get minimized as a robot may loose a task to another robot if it needs to take a detour when accepting the new task due to possible congestions or roadblocks.

3.3. Task-agent architecture

If the standard SSI-algorithm described in Section 3.1 is combined with the routing extension described in Section 3.2, the total auction algorithm with routing constraints at the AGV side (bidding side) is obtained. The AGV uses only local information to compute a bid: own location p, local task list LT, information on the newly announced task T_{new}, and the objective parameter ε. Only the graph G of the layout including all reservations could be seen as a global piece of information although each robot only accesses the part it needs in order to participate. Algorithm 2 shows the pseudo-code of our SSI-approach under routing constraints. The task-agent at AGV side outputs a decision, which is the total bid b it sends to the auctioneer. The 'DMAS()' function implements the delegate-MAS behavior as stated in Section 3.2.2.

The task-agent architecture at auctioneer side implements the winner determination algorithm defined in Section 3.1.2. Algorithm 3 shows the pseudo-code of this procedure.

Algorithm 2 AGV algorithm

1: **procedure** BIDCALCULATION(p, LT, T_{new}, G, ε)
2: $c_1 \leftarrow DMAS(p, LT, G)$
3: $LT' \leftarrow LT \bigcap T_{new}$
4: $c_2 \leftarrow DMAS(p, LT', G)$
5: $b_{ms} \leftarrow c_2 - c_1$
6: $b_{mm} \leftarrow c_2$
7: $b \leftarrow \varepsilon * b_{ms} + (1 - \varepsilon) * b_{mm}$
8: **return** b

Algorithm 3 Auctioneer algorithm

1: **procedure** AUCTION(T_{new}, R)
2: $announce(T_{new}, R)$
3: $bids \leftarrow receiveBids()$
4: $b_i \leftarrow min(bids)$
5: $assign(T_{new}, r_i)$

4. Illustrative example problem

Our proposed approach will be illustrated with a simple example. Assume a layout as shown in Fig. 1 is available. Costs (in seconds) are calculated as follows: $cost_{12} = d_{12}/v$, with $v = 1\ m/s$.

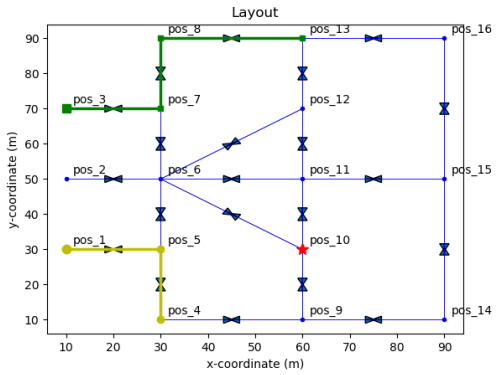


Fig. 1: Layout

Let us say there are two ($n = 2$) robots with initial locations for robots 1 and 2 respectively: $p_{R1} = pos_1$, $p_{R2} = pos_3$, and local task lists: $LT_{R1} = [pos_4]$ and $LT_{R2} = [pos_{13}]$. Both robots calculated following paths (yellow dots for robot 1 and green squares for robot 2) with the tasks to execute marked in bold:

$$path_{R1} = [pos_5, \boldsymbol{pos_4}]$$

$$path_{R2} = [pos_7, pos_8, \boldsymbol{pos_{13}}]$$

Both robots reserved following time slots in the environment for each of the nodes on their paths:

$$slots_{R1} = [(0, 20), (20, 40)]$$

$$slots_{R2} = [(0, 20), (20, 40), (40, 70)]$$

Both robots computed following costs for executing these paths:

$$cost_{R1} = 20s + 20s = 40s$$

$$cost_{R2} = 20s + 20s + 30s = 70s$$

Thus for example, robot 1 reserved node pos_5 for the time slot between 0 and 20 seconds after the robot starts executing the route. The costs are the costs of the current route they need to follow with the tasks they are already assigned to.

In the following, it will be demonstrated how our approach acts when a new task arrives at position pos_{10} (depicted with a red star in Fig. 1). It is also assumed that node pos_9 is blocked in a time slot (40, 55) due to a roadblock or due to another AGV passing that node in that particular time slot. After receiving the announced task from the auctioneer, both robots compute their bid for executing the new task on top of the tasks they are already assigned to. They do this considering routing constraints coming from a DMAS-algorithm. In this example, the total path cost is minimized by setting the tuning parameter in the objective function to one, $\varepsilon = 1$.

Robot 1: A DMAS-algorithm finds and evaluates the feasible routes for visiting both pos_4 and pos_{10}. A bid is calculated using this information.

- Feasibility phase, feasible paths:
  $path_{R1,1} = [pos_5; \boldsymbol{pos_4}; pos_9; pos_{10}]$
  $path_{R1,2} = [pos_5; \boldsymbol{pos_4}; pos_5; pos_6; pos_{10}]$
- Exploration phase, slots:
  $slots_{R1,1} = [(0, 20), (20, 40), \mathbf{(55,\ 85)}, (85, 105)]$

$slots_{R1,2} = [(0, 20), (20, 40), (40, 60), (60, 80), (80, 116)]$

- Exploration phase, total path cost:
  $cost_{R1,1} = 20s + 20s + \mathbf{15s} + 30s + 20s = 105s$
  $cost_{R1,2} = 20s + 20s + 20s + 20s + 36s = 116s$
  $\Rightarrow$ best path $= path_{R1,1}$

- Bid calculation:
  MiniSum bid: $b_{ms} = c_2 - c_1 = 105s - 40s = 65s$
  MiniMax bid: $b_{mm} = c_2 = 105s$
  Total Bid: $b_{R1} = 1 \cdot 65s + (1 - 1) \cdot 105s = \mathbf{65s}$

The slots that robot 1 would like to reserve, as well as the already reserved slots of robot 2 and the blockage at pos_9 are depicted in Fig. 2, which shows the environmental agent schedules for each node in the graph. Note that robot 1 cannot immediately access node pos_9 at time $t = 40s$. It needs to wait up to $t = 55s$ after which node pos_9 is unblocked (blockage is denoted by 'X'). The robot therefore reserves a slot $(55, 85)$ instead of $(40, 60)$ with a delay of 15s.

Robot 2: A DMAS-algorithm finds and evaluates the feasible routes for visiting both pos_{13} and pos_{10}. A bid is calculated using this information.

- Feasibility phase, feasible paths:
  $path_{R2,1} = [pos_7; pos_8; \mathbf{pos_{13}}; pos_{12}; pos_{11}; pos_{10}]$
  $path_{R2,2} = [pos_7; pos_8; \mathbf{pos_{13}}; pos_{16}; pos_{15}; pos_{11}; pos_{10}]$

- Exploration phase, slots:
  $slots_{R2,1} = [(0, 20), (20, 40), (40, 70), (70, 90), (90, 110), (110, 130)]$
  $slots_{R2,2} = [(0, 20), (20, 40), (40, 70), (70, 100), (100, 140), (140, 170), (170, 190)]$

- Exploration phase, total path cost:
  $cost_{R2,1} = 20s + 20s + 30s + 20s + 20s + 20s = 130s$
  $cost_{R2,2} = 20s + 20s + 30s + 30s + 40s + 30s + 20s = 190s$
  $\Rightarrow$ best path $= path_{R2,1}$

- Bid calculation:
  MiniSum bid: $b_{ms} = c_2 - c_1 = 130s - 70s = 60s$
  MiniMax bid: $b_{mm} = c_2 = 130s$
  Total Bid: $b_{R2} = 1 \cdot 60s + (1 - 1) \cdot 130s = \mathbf{60s}$

The slots that robot 2 would like to reserve, as well as the already reserved slots of robot 1 and the blockage at pos_9 are depicted in Fig. 3, which shows the environmental agent schedules for each node in the graph.

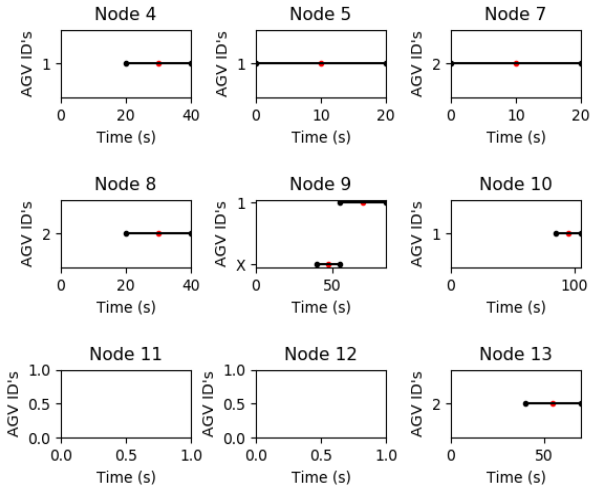


Fig. 2: Environmental agent schedules when robot 1 accepts the new task

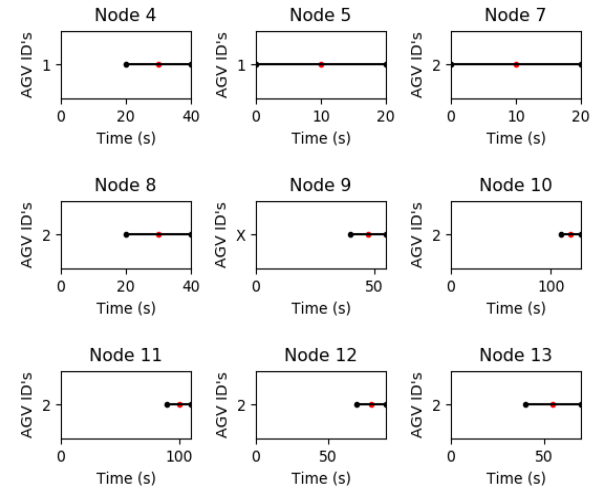


Fig. 3: Environmental agent schedules when robot 2 accepts the new task

It can be noticed that although robot 1 is closest to the new task, robot 2 can offer a lower bid and gets as-

signed the task because of the temporal blockage at node pos_9. When this assignment takes place, the total path cost (edge cost and delays) for both robots becomes $(40s + 130s) = 170s$ and the total execution time for executing all tasks (largest path cost of both robots) becomes $max(40s, 130s) = 130s$. The overall objective of this solution is:

$$obj = \varepsilon * 170s + (1 - \varepsilon) * 130s$$

And with our objective parameter $\varepsilon = 1$:

$$obj = 1 * 170s + (1 - 1) * 130s = 170s$$

This shows that considering routing constraints in the bidding process, in addition to preventing congestions, can reduce the task allocation objective, which is in this example the total path cost for all robots combined. In short, introducing routing constraints in the auction process offers a more realistic way of allocation considering dynamic changes in the environment that can cause delays in the routing when a new task would be accepted.

5. Validation and benchmarking

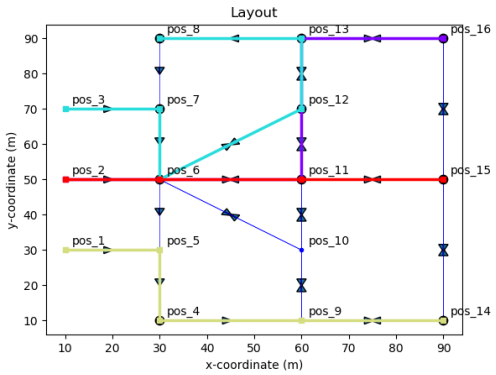
In this section, our approach is validated and benchmarked against an SSI-approach that does not use routing constraints in its bidding process. For validation, two stages are defined. First, the actual allocation of a random set of tasks is performed using the SSI-auction with (for our approach) and without (for the benchmark approach) including routing constraints. Second, the allocation result is evaluated by calculating the cost for all robots fully executing their allocated task sequence. This cost is calculated by computing routes for all robots through their assigned task sequences individually, either by using A-star or DMAS as a routing approach. In this evaluation, the number of congestions, the total delay, the total edge cost (in seconds), and the total path cost (edge cost plus delay), are calculated for all robots together. Three validation setups are defined:

- The first setup (SSI with simple bid and A-star routing), hereafter called setup 1, uses the classical sequential single-item auction process where bids are calculated as in Section 3.1.2 without considering routing constraints. The allocation is evaluated using the simple A-star algorithm for routing. This setup is meant to analyze the behavior of a robot that makes decisions based on a naive model of the world, both for task scheduling and routing. In other words, the robots do not consider possible collisions or route delays in the allocation process nor in the routing process. So when this plan of routes for each robot would be executed, collisions could occur. This is thus actually not a valid solution as robots do not consider other robots on the route, and could thus collide with each other when executing their tasks. This setup only has as a purpose to show that collisions will occur on the route, and will in comparison with setup 2, show that these collisions will be translated into delays by the DMAS-algorithm. Other static path planning methods than A-star, that do not consider possible delays on the route, could be used. However, they would show the same results, being that the congestions they cause due to no road segment allocations, will be translated into delays by the DMAS-routing approach.
- The second setup (SSI with simple bid and DMAS-routing), hereafter called setup 2, also uses the classical sequential single-item auction process where bids are calculated as in Section 3.1.2 without considering routing constraints. But here, the allocation is evaluated using the delegate-MAS algorithm for routing. This setup is meant to analyze the behavior of a robot that makes decisions based on a naive model of the world for task scheduling but not for routing. In other words, the robots do not consider possible collisions or route delays in the allocation process but do so in the routing process.

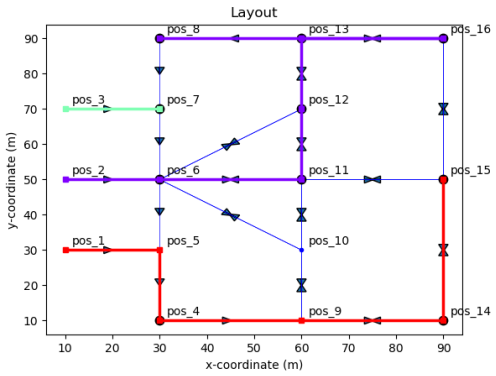
- The third setup (SSI with DMAS-bid and DMAS-routing), hereafter called setup 3, implements our approach and uses the sequential single-item auction process where bids are calculated as in Section 3.2 considering routing constraints. The allocation is evaluated using the delegate-MAS algorithm for routing. This setup is meant to analyze the behavior of a robot that makes decisions based on a more realistic model of the world for task scheduling as well as for routing. In other words, the robots do consider possible collisions and route delays both in the allocation process and in the routing process.

For all setups, the same layout as in Fig. 1 is used. All tasks are spawned at a random location and a task allocation is performed for each setup considering all tasks together without any time dependencies. The robots always start at their depot station. In all experiments and in all setups, the optimization objective is the minimization of the linear combination of the total travel cost and the total execution time.

In the simulations, the tune-able objective parameter ε is set to zero, thus minimizing the total execution time of executing all tasks. Simulated are done for one up to six robots and this for one up to ten tasks. Ten simulations are performed for each combination to analyze eventual stochastic variations. Fig. 4 shows some example assignment of all three setups (setup 1 and 2 always output the same task allocation solution as both bidding rules are the same). It can be seen that our approach maintains a better routing balance and attempts to avoid congestions by reducing the number of used robots and avoiding dense nodes that are already occupied. This all effectuates from within the bidding procedure. In the following, the results for our simulations are shown. Remark that only figures for three up to six robots are provided as possible congestions, and thus the effect of our approach, are low when only few robots move on the layout. Also remark that only figures for five up to ten tasks are provided for the same reason.



(a) Assignment of setup without DMAS in the bid procedure.



(b) Assignment of setup with DMAS in the bid procedure.

Fig. 4: Assignment example for a task allocation process of all setups.

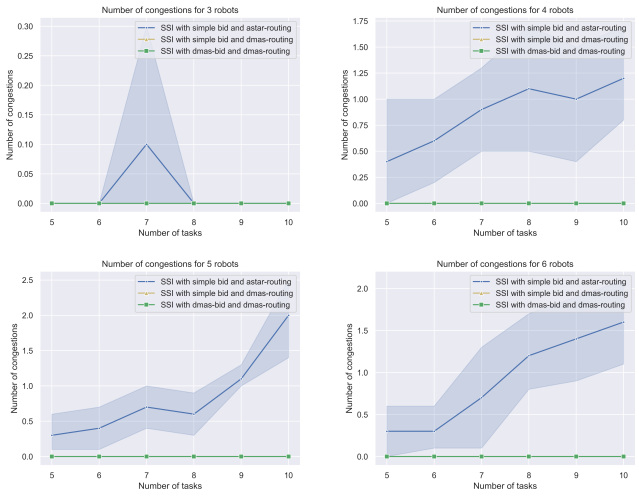


Fig. 5: Number of congestions in function of the number of tasks for three up to six robots

Fig. 5 shows the number of congestions that occurred when performing a task allocation and routing for each of the three setups and this for five up to ten tasks, and

three up to six robots. Remark that the curve of setup two coincides with that of setup three as both consider possible delays when routing and thus avoid congestions. From Fig. 5, it can be noticed that for situations with three up to six robots, congestions occur, increasing with the number of robots and number of tasks. However, this occurs only for setup 1 as this setup does not consider the possibility of collisions. For the other two setups, it can be seen that in the same situations, all congestions have been prevented thanks to applying the delegate-MAS as a routing approach after task scheduling. In order to avoid these congestions, those two setups took into account some delays in their routes in order to wait for an occupied road segment before it becomes free.

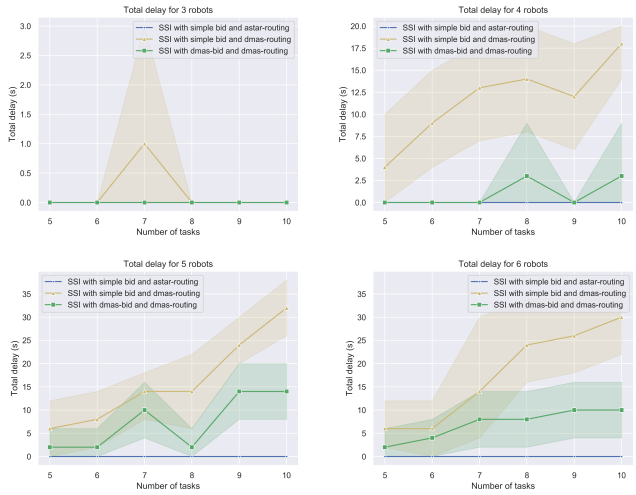


Fig. 6: Total delay in function of the number of tasks for three up to six robots

From Fig. 6 it is visible that setup two, which does not include routing constraints into its bidding process, clearly adds more delays into the routes than our approach, which does include routing constraints in its bidding process. This is because our approach already takes routing constraints into account when considering to accept a new task. Also remark the same trends of the congestions in Fig. 5 and the delays in Fig. 6. This similarity is obvious as only delays are needed if otherwise a congestion could occur. The delay graph of setup two literally follows the congestion trend of setup one as both output exactly the same task allocation result, resulting in the same routes for all robots. The only difference is that setup two adds delays to these routes where setup one came into collision.

Fig. 7 shows that the setup that implements our approach always outputs a lower total edge cost than the other two setups that both output the same edge cost as they have the same task allocation result and thus the same routes. This results from the routing balancing effect of our approach that reduces the number of robots used. The total effect of reducing delays and reducing the edge cost is visible in Fig. 8 where both effects add up. It can be seen that our approach always outputs a lower total path cost summed over all robots. This effect is also remarkable with other values for the objective parameter.

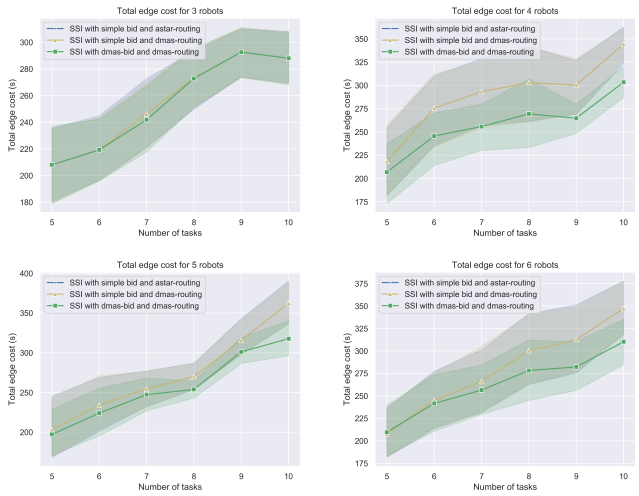


Fig. 7: Total edge cost in function of the number of tasks for three up to six robots

6. Conclusions

In this paper, a decentral task allocation architecture for a fleet of mobile robots is presented based on the sequential single-item auction principle considering routing constraints. As a routing strategy, a resource allocation approach based on ant colony optimization processes called delegate-MAS is used. It can be concluded that our approach is capable of eliminating congestions due to the allocation of resources in the virtual guide path network,

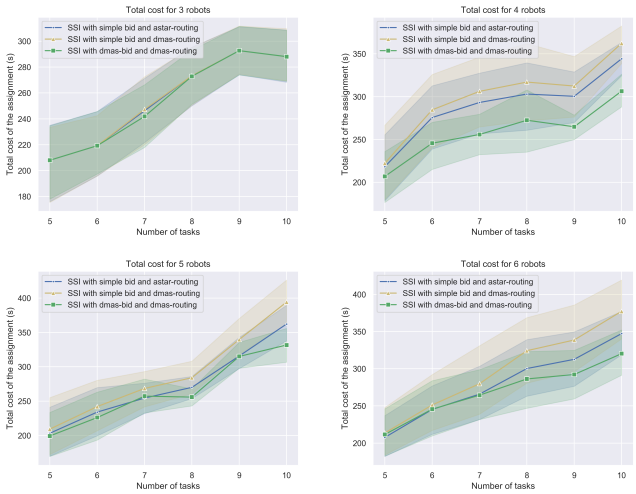


Fig. 8: Total path cost in function of the number of tasks for three up to six robots

and results in a more realistic routing model that is used in the task allocation process. The implementation of routing information in the task allocation process can easily be done by just adding the extra routing cost in the bidding procedure of the auction process.

Our approach causes robots to consider the possibility of being delayed when accepting a new task due to roadblocks or occupied nodes, and could thus be less suitable for executing the new task. This causes that possible routing delays get minimized and that the network load gets more balanced. As a result, the possibility of congestion or deadlock occurrence could be reduced. However, our approach will not always guarantee a fully deadlock-free system, but deadlock prevention and deadlock resolution algorithms could be added to our proposed algorithm in future work.

In our approach, we used the delegate-MAS algorithm for dynamic multi-robot routing. However, there are several other routing algorithms that could be used as a routing strategy of which the estimated delays are used in the task allocation process. For future work, it could be interesting to investigate the possibility of Machine Learning methods in order to solve the routing problem. In future research, also uncertainty in the bidding process could be introduced. In this paper, it is assumed that path costs are deterministic. However, in reality, path costs can be uncertain due to crowdedness on the paths or due to obstacles on the path. The cost to traverse a certain path could be modeled by using an imprecise uncertainty model [31, 32] that considers the known uncertainty on path cost in function of another factor like daytime (morning, rush period, evening).

Acknowledgments

This work is supported by the M-group, part of the KU Leuven Campus in Bruges.

References

[1] I. Draganjac, D. Miklic, Z. Kovacic, G. Vasiljevic, S. Bogdan, Decentralized Control of Multi-AGV Systems in Autonomous Warehousing Applications, IEEE Transactions on Automation Science and Engineering 13 (4) (2016) 1433–1447. doi:10.1109/TASE.2016.2603781.

[2] M. P. Fanti, A. M. Mangini, G. Pedroncelli, W. Ukovich, A decentralized control strategy for the coordination of AGV systems, Journal of Control Engineering Practice 70 (2018) 86–97. doi:10.1109/CoDIT.2016.7593609.

[3] I. Baffo, G. Confessore, G. Stecca, A decentralized model for flow shop production with flexible transportation system, Journal of Manufacturing Systems 32 (1) (2013) 68. doi:10.1016/j.jmsy.2012.10.002.

[4] M. De Ryck, M. Versteyhe, F. Debrouwere, Automated guided vehicle systems, state-of-the-art control algorithms and techniques, Journal of Manufacturing Systems 54 (2020) 152–173. doi:10.1016/j.jmsy.2019.12.002.

[5] M. De Ryck, D. Pissoort, T. Holvoet, E. Demeester, Decentral task allocation for industrial AGV-systems with resource constraints, Journal of Manufacturing Systems 59 (2021) 310–319. doi:10.1016/j.jmsy.2021.03.008.

[6] M. De Ryck, M. Versteyhe, K. Shariatmadar, Resource management in decentralized industrial Automated Guided Vehicle systems, Journal of Manufacturing Systems 54 (2020) 204–214. doi:10.1016/j.jmsy.2019.11.003.

[7] H. T. Dinh, R. R. S. V. Lon, T. Holvoet, Multi-Agent Route Planning Using Delegate MAS, in: ICAPS Proceedings of

the 4th Workshop on Distributed and Multi-Agent Planning (DMAP-2016), 2016, pp. 24–32.

[8] S. Hanif, R. R. Van Lon, N. Gui, T. Holvoet, Delegate MAS for large scale and dynamic PDP: A case study, Studies in Computational Intelligence 382 (2011) 23–33. doi:10.1007/978-3-642-24013-3_4.

[9] B. Micieta, M. Edl, M. Krajcovic, L. Dulina, P. Bubenik, L. Durica, V. Binasova, Delegate MASs for coordination and control of one-directional AGV systems: a proof-of-concept, International Journal of Advanced Manufacturing Technology 94 (1-4) (2018) 415–431. doi:10.1007/s00170-017-0915-8.

[10] Q. Li, J. T. Udding, A. Pogromsky, Zone-control-based traffic control of automated guided vehicles, Lecture Notes in Control and Information Sciences 456 (2015) 53–60. doi:10.1007/978-3-319-10407-2_7.

[11] J. Zając, W. Małopolski, Structural on-line control policy for collision and deadlock resolution in multi-AGV systems, Journal of Manufacturing Systems 60 (December 2020) (2021) 80–92. doi:10.1016/j.jmsy.2021.05.002.

[12] M. P. Fanti, A deadlock avoidance strategy for AGV systems modelled by coloured Petri nets, Proceedings - 6th International Workshop on Discrete Event Systems, WODES 2002 (2002) 61–66doi:10.1109/WODES.2002.1167670.

[13] R. Lochana Moorthy, W. Hock-Guan, N. Wing-Cheong, T. Chung-Piaw, Cyclic deadlock prediction and avoidance for zone-controlled AGV system, International Journal of Production Economics 83 (3) (2003) 309–324. doi:10.1016/S0925-5273(02)00370-5.

[14] E. Roszkowska, S. A. Reveliotis, On the liveness of guidepath-based, zone-controlled dynamically routed, closed traffic systems, IEEE Transactions on Automatic Control 53 (7) (2008) 1689–1695. doi:10.1109/TAC.2008.929375.

[15] A. Krnjak, I. Draganjac, S. Bogdan, T. Petrovic, D. Miklic, Z. Kovacic, Decentralized control of free ranging AGVs in warehouse environments, Proceedings - IEEE International Conference on Robotics and Automation 2015-June (June) (2015) 2034–2041. doi:10.1109/ICRA.2015.7139465.

[16] M. Jäger, B. Nebel, Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots, IEEE International Conference on Intelligent Robots and Systems 3 (2001) 1213–1219. doi:10.1109/IROS.2001.977148.

[17] V. Digani, L. Sabattini, C. Secchi, C. Fantuzzi, Towards decentralized coordination of multi robot systems in industrial environments: A hierarchical traffic control strategy, in: Proceedings - 2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing, ICCP 2013, IEEE, 2013, pp. 209–215. doi:10.1109/ICCP.2013.6646110.

[18] L. Pallottino, V. G. Scordio, A. Bicchi, E. Frazzoli, Decentralized cooperative policy for conflict resolution in multivehicle systems, IEEE Transactions on Robotics 23 (6) (2007) 1170–1183. doi:10.1109/TRO.2007.909810.

[19] S. A. Reveliotis, E. Roszkowska, Conflict resolution in free-ranging multivehicle systems: A resource allocation paradigm, IEEE Transactions on Robotics 27 (2) (2011) 283–296. doi:10.1109/TRO.2010.2098270.

[20] E. Steegmans, T. Holvoet, N. Janssens, S. Michiels, E. Berbers, P. Verbaeten, P. Valckenaers, H. Van Brussel, Ant Algorithms in a Graph Environment: a Meta-scheme for Coordination and Control, Artificial Intelligence and Applications (January) (2002) 435–440.
URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.70.9357

[21] E. Nunes, M. Gini, Multi-robot auctions for allocation of tasks with temporal constraints, Proceedings of the National Conference on Artificial Intelligence 3 (2015) 2110–2116.

[22] E. Nunes, M. McIntire, M. Gini, Decentralized multi-robot allocation of tasks with temporal and precedence constraints, Advanced Robotics 31 (22) (2017) 1193–1207. doi:10.1080/01691864.2017.1396922.

[23] A. Schoenig, M. Pagnucco, Evaluating sequential single-item auctions for dynamic task allocation, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6464 LNAI (2010) 506–515.

[24] S. Koenig, C. Tovey, M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, A. Meyerson, S. Jain, The power of sequential single-item auctions for agent coordination, Proceedings of the National Conference on Artificial Intelligence 2 (2006) 1625–1629.

[25] A. Farinelli, N. Boscolo, E. Zanotto, E. Pagello, Advanced approaches for multi-robot coordination in logistic scenarios, Robotics and Autonomous Systems 90 (2017) 34–44. doi:10.1016/j.robot.2016.08.010.

[26] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, S. Jain, Auction-based multi-robot routing, Robotics: Science and Systems 1 (June) (2005) 343–350.

[27] R. Matai, S. Singh, M. Lal, Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches, in: Traveling Salesman Problem, Theory and Applications, no. January 2014, 2010. doi:10.5772/12909.

[28] C. Liu, A. Kroll, A centralized multi-robot task allocation for industrial plant inspection by using A* and genetic algorithms (2012). doi:10.1007/978-3-642-29350-4_56.

[29] N. Sullivan, S. Grainger, B. Cazzolato, Sequential single-item auction improvements for heterogeneous multi-robot routing,

Robotics and Autonomous Systems 115 (2019) 130–142. doi:10.1016/j.robot.2019.02.016.
URL https://doi.org/10.1016/j.robot.2019.02.016

[30] T. Vidal, J. Bidot, Dynamic sequencing of tasks in simple temporal networks with uncertainty, CP 2001 Workshop in Constraints and Uncertainty (2001) 1–10.
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.130.313&rep=rep1&type=pdf

[31] K. Shariatmadar, M. D. Ryck, F. Debrouwere, M. Versteyhe, CMMSE-Decentralised Automated Guided Vehicle Systems under uncertainty, Cmmse (September) (2019).

[32] K. Shariatmadar, M. Versteyhe, Linear programming under p-box uncertainty model, 2019 IEEE 7th International Conference on Control, Mechatronics and Automation, ICCMA 2019 (2019) 84–89doi:10.1109/ICCMA46720.2019.8988632.