Decentral Task Allocation for Industrial AGV-Systems with Resource 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, especially in Europe. Today's AGV-systems offered by global manufacturers almost all operate under some form of centralized control where a single central controller coordinates the entire fleet of AGVs. There is a trend towards decentralized control of these systems where AGVs make individual decisions that promote the flexibility, robustness and scalability of transport. However, its practical implementation seems to be in its infancy. In addition to the lack of practical implementation of decentralized control in industrial AGV-systems, we have observed a research gap in intelligent resource management of AGV-systems, which we have tried to address in previous work by proposing a more intelligent resource management approach. In this paper, we have addressed both the perceived lack of practical decentralized AGV control and the lack of intelligent resource management by proposing a decentralized task allocation algorithm based on sequential single-item auctions, taking into account resource constraints, and in which our intelligent resource management approach from previous work is introduced. We have benchmarked our new approach to a genetic algorithm-based task-allocation solver that uses "threshold-100"-charging as a resource management strategy. The result of the proposal is a decentralized task-allocation architecture under resource constraints that could be used in current AGV-systems to add more decentralized features to the fleet.

Keywords: Automated Guided Vehicles, Decentralization, Task Allocation, Resource 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. In many cases, an entire fleet of mobile robots cooperates to efficiently transport goods. These systems are typically all controlled in a centralized manner where one single computer coordinates the whole fleet, has global information, and uses optimization-based heuristics to find global solutions [1–3]. This central idea works well for small and simple systems but it lacks flexible manufacturing paradigms about (i) robustness: maintaining a high fault tolerance against uncertain environments (road blockage, device malfunction, etc.), (ii) flexibility: being able to dynamically adapt to changing circumstances, and (iii) scalability: having a performance that is invariant to the scale of the system.

*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)

Decentralization of control can provide solutions to these drawbacks. Literature proves the benefit of decentralized over centralized control and states the need for it [4–6]. Research on decentralized control architectures is very extensive and many algorithms are elaborated and tested very deeply, and show promising results [7, 8]. But still, this innovative technology has not fully found its way to industrial practice because decentralized control is generally complex to implement. Many individual entities need to coordinate and need to make decisions based on solely local information. In our research, we aim at facilitating the gradual implementation of decentralized control by investigating a migration methodology from centralized towards decentralized control. We attempt to do this by decomposing the total AGV-system into its components which is a known principle for reducing complexity in e.g. computer science and is often called "Divide and Conquer". In [9], we have divided the control system of an AGV-fleet into five distinct tasks: Task Allocation, Localization, Path Planning, Routing, and Resource Management. In this paper, we have targeted the task allocation problem using auction-based principles, more particularly the sequential single-item auctions. Besides the task allocation, we have also considered resource constraints by introducing our previous work that focused on the resource management problem within an AGV-system. In this previous work [10], we have proposed an intelligent resource management approach in which an AGV chooses an optimal insertion of a charging station in its initial sequence of tasks as well as an optimal charging time needed to finish its task sequence with zero (or a predefined minimum of) resources (zero battery level). The result from this previous research is included in the task allocation process by means of a possible extra cost of charging in the bidding mechanism. By doing this, AGVs bid on tasks considering the possibility of the need to charge when accepting a new task. This results in a task allocation that uses a more realistic behaviour of the real AGVs, namely the fact that they all have a limited amount of resources (remaining battery power). The fusion of our earlier presented decentral resource management approach and our task allocation approach presented in this paper requires very few adaptions to the control architecture, facilitating their implementation in industrial AGV-systems.

The paper is organized as follows: Section 2 covers the literature on task allocation and refers to our previous work on resource management. Section 3 covers the proposed decentral task allocation architecture under resource 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

The research on task allocation for multi-robot systems is very extensive [11–13]. Task allocation algorithms aim at allocating a set of tasks $T = \{t_1, t_2, \ldots, t_m\}$, to a set of robots $R = \{r_1, r_2, \ldots, r_n\}$ in the most optimal way regarding one or more specific objectives. In [14], a complete taxonomy on task allocation problems is provided which is frequently used in task allocation literature. In literature, two main approaches to solve the task allocation problem can be distinguished: optimization-based approaches, and market-based approaches [15].

2.1. Optimization-based approaches

Optimization-based approaches are mostly used in centralized control systems where one controller has access to global information and tries to find a global optimal allocation of tasks. When the number of tasks and the number of robots increases, the number of possible allocations scales factorial making exact search algorithms unusable. Such an NP-hard combinatorial optimization problem is mostly solved using heuristics or

meta-heuristics. Some commonly used heuristic optimization algorithms for optimization-based task allocation are: Genetic Algorithms [16, 17], Simulated Annealing [18], and Ant Colony Optimization [19]. In [9], we made a broader review on the algorithms and techniques of optimization-based approaches. Task allocation based on this approach is optimal for small and simple systems but lacks optimality for larger and more complex systems as the computational complexity increases exponentially with the number of robots and with the number of tasks. In general, the number of iterations of an optimization-based solver are limited by the user to bound the total execution time of the optimization process in order to compute more task allocations at a time, allowing a faster response to dynamic changes in the environment. The more the optimization time is limited, the fewer solutions can be explored by the algorithm. This prevents the heuristic algorithm of fully exploring the search space, resulting in bad optimization performance. This causes the task allocation of larger and more complex systems to be highly sub-optimal. Alternatively, decentralized market-based task allocation approaches could be used.

2.2. Market-based approaches

Market-based (or auction-based) approaches can overcome the limitations of optimization-based approaches as they have computational times that scale better with the problem complexity. The efficiency of auction-based methods has been demonstrated experimentally in [20, 21]. Market-based approaches are decentralized optimization approaches that use an auction process with bids to obtain an assignment. In general, an auctioneer announces a task to all robots in the system, which in response calculates a bid on the task and sends it back to the auctioneer. The auctioneer in turn assigns the task to the robot with the lowest bid. This most basic auction-based principle is called the CNET-protocol [22]. There exist many variations to this protocol which are deeply investigated in literature and which are discussed further in this section.

2.2.1. Auction principles

Market-based approaches can be roughly divided into three main principles: parallel single-item auctions, combinatorial auctions, and sequential single-item auctions [23]. In the following, it is assumed that several tasks need to be allocated at once.

Parallel single-item auction. In parallel single-item auctions, each robot calculates a bid for each of the tasks and the auctioneer assigns all tasks at once. The computational complexity of this protocol is $O(n_{robots}.n_{tasks})$ but the solutions are likely to be highly sub-optimal since it does not take any synergies between tasks into account [23].

Combinatorial auction. In combinatorial auctions, each robot calculates a bid for every subset of the tasks on offer. By considering every possible combination of tasks in the bidding process, synergies between tasks are taken into account. In combinatorial auctions, $2^n - 1$ bids are required when n tasks are on offer. The computational complexity is $O(2^{n_{tasks}}.n_{robots})$ [23]. Combinatorial auctions generally produce close to optimal solutions but need a high computation time in return.

Sequential single-item auction. Sequential single-item auctions (marked as SSI-approach in the remainder of the paper) are a compromise between parallel and combinatorial auctions if it comes to optimality and computational effort. The auction proceeds over several rounds and one task is assigned to a robot in each round. Sequential single-item auctions are in general not guaranteed to find the optimal solution. However, they clearly provide better solutions compared to parallel auctions as synergies between tasks can be exploited. The computational complexity of this protocol is $O(n^2_{tasks}.n_{robots})$, which is a significant improvement over combinatorial auctions [23].

Literature [11, 23–25] shows that sequential single-item auctions have the best trade-off between optimality and simplicity.

2.2.2. Auction phases

The total auction process can be divided into three phases: an initial phase, a bidding phase, and a winner determination phase. For each of them, we covered the most popular principles.

Initial phase. In the initial phase, an auctioneer announces arriving tasks to the robots. This phase is divided into two aspects: the auctioneer's role, and the region of announcement.

- Auctioneer's role: The auctioneer can be either centralized or decentralized. When the auctioneer is centralized, this means that all tasks arrive at this same auctioneer and this one announces all tasks to the robots in the system. The auctioneer can be a separate piece of hardware or can be one of the robots. In a decentralized auctioneer's role, there can be multiple auctioneers in parallel. These multiple auctioneers can be some fixed predefined robots or the auctioneer roles can alter between robots following a certain alternation principle. The alternation of the auctioneer roles between the group of robots can, for example, be time-based or token-based.

- Region of announcement: This is generally to be considered for large scale systems. When a task arrives at an auctioneer, the auctioneer can announce the task to all robots in the system. For very large systems, this asks for many unnecessary computations as all robots, even those far away from the task, need to compute bids. Another solution is to announce the task to a subset of robots. A task can be announced to all robots in a certain region around the auctioneer. Or the task can be assigned to all robots in a certain region around the spawned task's location.

Literature on the effect of different auctioneer roles and to the effect of different regions of announcement is very sparse. Some papers [15, 26] mention the possibility of centralized and decentralized auctioneers shortly but as far as the authors know, no one compared the influence of different auctioneer roles or different regions of announcement on the system's performance experimentally.

Bidding phase. In the bidding phase, robots evaluate the tasks and calculate the cost to execute the tasks they are interested in. Bidding rules are used by the individual robots to calculate the cost to execute a particular task and compute it as a bid to send to the auctioneer for selection. The bidding rules depend on the objective that holds. For allocation, multiple objectives can be considered [27]:

- MiniSum: Minimize the sum of robot path costs over all robots

- MiniMax: Minimize the maximum robot path cost over all robots

- MiniAve: Minimize the average task path cost over all tasks

The robot path cost is the total cost for the robot to execute all the tasks it is assigned to. The task path cost is the total cost of the traversed path of a task by the robot it is assigned to. The first one causes the fleet to assign a task in a way that minimizes the total travel cost, and thus also the consumed energy. However, this can cause some robots to execute all tasks in their neighbourhood while other robots may be idle for a time, causing the execution of all tasks to take longer than necessary. Here, the MiniMax objective comes into play which maximizes the total AGV usage and causes all tasks to be executed in a minimal time span. MiniAve can be used when it is not appropriate to have a parcel on travel for a long period.

To achieve the mentioned objectives, three main bidding rules exist [27]:

- MiniSum: This bidding rule achieves the MiniSum objective. Each robot bids the extra cost for executing a new task besides the already allocated tasks. This is called the marginal cost. This causes tasks to be allocated to the robots that can complete them with the least extra cost.
- MiniMax: This bidding rule achieves the MiniMax objective. Each robot bids the total cost of completing both the current allocated tasks and the new task. This causes tasks to be allocated somewhat evenly between all robots.
- MiniAve: This bidding rule achieves the MiniAve objective. Each robot bids the average cost for each of its assigned tasks to go from the initial robot location, over the pick-up location, to the drop-off location. This causes tasks to be at their destination as fast as possible.

Winner determination phase. In the winner determination phase, the auctioneer determines the winner for each of the tasks and notifies the winning robots. However, in sequential single-item auctions, the auctioneer only assigns one task at a time out of a list of tasks to be allocated. It collects the bids of every robot on all of the announced tasks, compares them, and selects only one task to assign to the winning robot of that task following a certain resolution rule. All other remaining tasks are auctioned again. The rule of selecting a bid can alter [12]:

- Lowest Bid (LB): The auctioneer allocates the task with the overall lowest bid to the bidder of that lowest bid. Robots must submit only their lowest bid to the auctioneer. This causes the total cost to be minimized.
- Biggest Bid Difference (BD): For each task in the tasks to allocate, the auctioneer looks at the minimum and maximum bid it received from the robots for that particular task. Out of the tasks to allocate, the auctioneer allocates the task with the largest difference between these minimum and maximum bids on that task. The robot that submits the lowest bid to that selected task is assigned the task. This to prioritize early allocation of tasks that some robots are unable or ineffective at completing.
- Fewest Bid (FB): The auctioneer allocates the task which received the fewest bids, e.g. when some robots do not bid on the task. The robot that submits the lowest bid to that task is assigned the task. This is used to prioritize early allocation of tasks that few robots are capable of completing. This is identical to Lowest Bid if robots are homogeneous (can execute all types of tasks).

2.3. Resource management

In earlier work [10], we presented an intelligent resource management approach for decentral industrial AGV-systems. In this approach, an AGV has a local list of tasks it needs to execute. Given a graph of the warehouse layout and its current battery level, the AGV computes the most optimal charging point in its initial sequence of tasks as well as an optimal charging time. Using this approach, the AGV finds that the most optimal charging point is the one closest to its initial path and that the optimal charging time is the time in which the AGV gains exactly the amount of resources to end all tasks with zero (or a specified minimum of) resources. We assume that any charging location is available for each robot at any time. This resource management strategy lends itself well to the implementation in a decentral auction-based task allocation approach as the extra time cost for driving to the charging station and the time cost to charge, can be included in the bid calculation of the auction process. As a result, when a new task arrives, AGV 1 can get the task rather than AGV 2 which needs

to charge if it would accept the task. And this even if the task is much closer to AGV 2. In the next sections, our novel task allocation architecture including this optimal charging behaviour is described.

3. Proposed task allocation architecture under resource constraints

3.1. Task allocation approach

Our proposed decentral task allocation approach is based on the sequential single-item approach as this is proven to have the best trade-off between optimality and simplicity and considers robots as bidders, and tasks as goods. In this section, we elaborate on the sequential single-item auction process and how we implemented it using a linear combination of two bid principles: MiniSum and MiniMax. In the subsequent section, we introduced our contribution to this approach using an optimal resource management scheme.

3.1.1. Setup

We have a set of tasks $T = \{t_1, t_2, \ldots, t_m\}$ which needs to be allocated and a set of robots $R = \{r_1, r_2, \ldots, r_n\}$ which can be used to execute the tasks. Tasks are considered to be purely transportation tasks in which a robot should pick up or drop something at the specified task location. Robots can have multiple tasks they are assigned to. Each robot maintains a local task list LT_i which is initially empty. Our proposed architecture is a decentralized architecture, this means that most of the knowledge is distributed. To participate into the auctions, each robot only needs local information: its own location, local task list, information on the newly announced task, and a graph of the layout. 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 $V = D \cup S$, and edges E which is a set of edges joining any two vertices from V.

3.1.2. Auction process

All tasks are initially unallocated and are announced for auction by an auctioneer once available. After receiving the announcement of a new task T_{new} from an auctioneer, all robots compute their bid for the task following some bidding rule which is covered in the next sections. 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 popular A* shortest path planning algorithm [3] and the vehicle's velocity v. After bid computation, all robots send their bids to the auctioneer which determines the winning robot following some winner determination rule covered in the next sections. The winning robot adds the task to its local task list LT_i using an optimal insertion heuristic [28] and takes the first task in its local task list to execute.

Initial phase. In this paper, we considered a central auctioneer which initiates each bidding procedure. This is one single entity that announces a new task to all robots in the system, receives all their bids, and allocates the task to the robot with the winning bid. Using a central auctioneer offers less decentralized characteristics than a decentralized auctioneer where the auctioneer's role is assigned to one of the robots in the fleet each time a new task arrives. The alternation of robots taking up this role can be coordinated using a token- or time-based schedule. However, the authors believe that implementing a centralized auctioneer is a first step towards practical decentralization as this offers some more controllability and predictability to the fleet in comparison with decentral auctioneers. It should be noted that using a decentralized auctioneer does not necessarily mean that the auctioneer's functions need to run on a separate piece of hardware. Also for the total auction process, this does not mean that all bidding algorithms need to run at a separate piece of hardware. These can all perfectly run on one physical entity. In any case, the choice between a central or a decentral auctioneer is totally independent

of the bidding rules adopted by the individuals which are described next.

Bidding phase. In our approach, we use a combination of the MiniSum and the MiniMax team objective by calculating the MiniSum and MiniMax bids, b_{ms} and b_{mm} for each robot. The MiniSum bidding rule causes a minimum total path cost over all robots. The MiniMax bidding rule causes a minimum total time span in which all tasks are executed. Both of them can have their benefits in certain situations. For this reason, we propose to include an extra parameter $\varepsilon \in [0, 1]$ which is tuneable 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:

$$b_{total} = \varepsilon \cdot b_{ms} + (1 - \varepsilon) \cdot b_{mm} \tag{1}$$

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 very urgently. 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 charge most of the robots and having few robots executing tasks. When the user wants both objectives to be minimized in an equal amount, ε can be put to 0.5.

To calculate the MiniSum bid b_{ms}, the robot needs to calculate the extra cost to insert the new task T_{new} in its local task list LT_i. First, it computes the total cost c_1 for executing its initial task list LT_i. Second, it needs to compute the total cost c_2 for executing a new local task list:

$$LT_i' = LT_i \bigcap T_{new} \tag{2}$$

which is the old local task list LT_i in which the new task T_{new} is inserted using an optimal insertion heuristic. Finally, it computes the difference of both costs:

$$b_{ms} = c_2 - c_1 \tag{3}$$

to obtain the MiniSum bid. To calculate the MiniMax bid b_{mm}, the robot needs to only calculate the total cost c_2 as in this approach, the robot only bids the total cost of executing the new task, so no marginal cost is calculated. When both bids are calculated, this is combined using equation 1 to obtain the final bid which is sent to the auctioneer for the winner determination phase.

Winner Determination phase. As a winner determination phase, we adopt the simple Lowest Bid (LB) principle. The auctioneer receives all bids from the robots at each task and assigns the task which has the lowest bid to the robot which computed that bid. We use this principle 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. Resource management approach

When resource constraints are considered and our approach for the resource management AGV task from earlier work is introduced, only minor adaptions need to be made to both the setup and the auction process.

3.2.1. Setup

In order to consider resource constraints, each robot has a charging level β which is denoted in % going from 0 to 100%. With resources, we mean the battery level of the robot, and thus the amount of energy it has still left. A resource consumption characteristic is defined as in [10]:

$$deltaR_c(t) = f(t) \tag{4}$$

Where the consumption $deltaR_c$ (battery loss) is a function of the travel time. The function $f(t)$ is dependent of the speed, accelerations, loading and unloading of the vehicle. After a certain consumption, the new battery level is:

$$\beta_{new} = \beta_{old} - deltaR_c \tag{5}$$

In addition, the robot also needs a list of charging stations where it is able to charge. Mostly, these charging stations are situated at the depot stations. So also in this approach, we assume that at each depot station d, any AGV can charge simultaneously. A charging characteristic can be defined as in [10]:

$$deltaR_{ch}(t) = (\frac{R_{max}}{t_{max}}) \cdot t \tag{6}$$

Where the charged amount is linearly dependent on the charging time.

3.2.2. Auction process

In order to consider resource constraints, this only needs little adaption to the bid calculation explained in Section 3.1.2. The only modification is in the calculation of c_2. For calculating c_2, the AGV needs to check whether it needs to charge when the new task T_{new} is optimally included in its local list using an optimal insertion heuristic. If the AGV can complete the new list LT_i' with the new task without charging, no adaptations need to be done to the bid calculation. If the AGV cannot complete the new list due to resource constraints, it can insert an optimal charging point d_{opt} as an extra task in its task list, which now becomes:

$$LT_i'' = LT_i' \bigcap d_{opt} \tag{7}$$

It also chooses an optimal charging time t_{opt}. The total cost c_2 now becomes the cost of executing local task list LT_i'' plus the charging time t_{opt}.

3.3. Task-Agent architecture

As we have two entities in market-based task allocation: auctioneer and robots, we introduce two task-agents architectures. One which adopts the bidding rule algorithm (the task-agent at AGV side), and one which adopts the winner determination principle (the task-agent at auctioneer side). A task-agent is defined as an intelligent agent inside the AGV-agent which takes care of a particular AGV task, which in this case is the task allocation AGV task. In the next sections, we defined the task-agent architectures on both the AGV side and the auctioneer's side.

3.3.1. Task allocation-agent at AGV side

The AGV uses its local information to compute a bid: own location p_i, local task list LT_i, information on the newly announced task T_{new}, a graph G of the layout, the battery level β, and the objective parameter ε. Algorithm 1 shows the pseudo code of this approach. The task-agent outputs a decision which is the total bid b it sends to the auctioneer.

Algorithm 1 AGV's algorithm

```
procedure BidCalculation(p, LT, T_new, G, β ε)
    c_1 ← previousCalculatedCost()
    LT' ← optimalInsert(LT, T_new, G)
    d_opt, t_opt ← optimalCharging(LT', β, G)
    LT'' = LT' ⋂ d_opt
    c_2 ← calculateCost(p, LT'', t_opt, β, G)
    b_ms ← c_2 − c_1
    b_mm ← c_2
    b ← ε · b_ms + (1 − ε) · b_mm
    return b
```

In this paper, we consider the 'calculateCost()' function to simply calculate the travel cost using an A* path planning algorithm. The 'optimalCharging()' function checks whether it is needed to charge and calculates an optimal charging point and optimal charging time when it does need to charge. In the future, the bid calculation can contain more complex information like routing information which considers extra costs to avoid

collisions and deadlocks, and uncertainty information which considers the stochastic nature of the path costs in time. If no resource management needs to be considered in the task allocation process, the line of calculating an optimal insertion point and charging time could easily be omitted. This can be the case when for example another, simpler, resource management scheme is followed that runs independently of the task allocation process.

3.3.2. Task allocation-agent at auctioneer side

The task allocation-agent architecture at auctioneer side implements the winner determination algorithm defined in Section 3.1.2. Algorithm 2 shows the pseudo code of this procedure.

Algorithm 2 Auctioneer's algorithm

```
procedure AUCTION(T_new, R)
    announce(T_new, R)
    bids ← receiveBids()
    b_i ← min(bids)
    assign(T_new, r_i)
```

4. Illustrative example problem

We illustrated the proposed approach with an example in which we show how it acts when some new task arrives in a situation of two robots each having their local task list of tasks they need to execute as well as a certain initial battery level. Assume we have a layout as shown in Fig. 1 with all distances in meters. Also assume that we have two ($n = 2$) robots with initial locations and battery levels for robots 1 (green squares) and 2 (yellow squares), respectively: $p_1 = pos_3, \beta_1 = 100\%$ and $p_2 = pos_1, \beta_2 = 45\%$. We assume one charging station (blue triangle) at location pos_2 and we arbitrarily assume that the battery consumption is linearly dependent on the travel time $\beta = -0.5 \cdot t$ as is the charging characteristic $\beta = 5 \cdot t$. The robots have local task lists:

$$LT_1 = [pos_7; pos_8; pos_13]$$

$$LT_2 = [pos_5; pos_4; pos_9]$$

In the following, we see how the approach acts when a new task T_{new} (red star) arrives at location pos_14. Costs (in seconds) are calculated as follows: d/v, with d a distance between two points in the layout, and $v = 1\ m/s$. We want to minimize the total travel time cost of the assignment as well as the maximum time span of the assignment. Thus we set the objective parameter ε at 0.5. An auctioneer (either centralized or decentralized) announces the task to the two robots. Every robot calculates its bid following the proposed bidding rule.

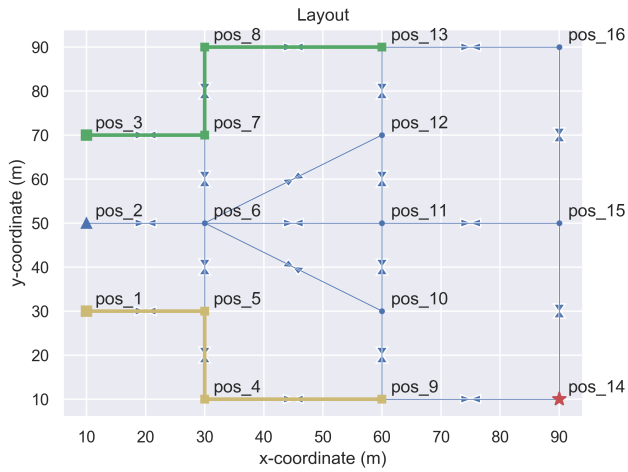


Fig. 1: Layout

- **Robot 1**:

  Local task list:

$$LT_1 = [pos_7; pos_8; pos_13]$$

  Cost starting from initial position pos_3:

$$c_1 = 20 + 20 + 30 = 70s$$

  Optimal insertion of task T_{new}:

$$LT_1' = [pos_7; pos_8; pos_13; \mathbf{pos_14}]$$

New cost starting from initial position:

$$c_2 = 20 + 20 + 30 + 30 + 40 + 40 = 180s$$

Battery check:

$$\beta_1 = 100\% - (0.5 \cdot 180)\% = 10\%$$

This robot can execute the extra task without the need of charging. Thus it's cost c_2 remains.
MiniSum bid: $b_{ms} = c_2 - c_1 = 180 - 70 = 110$
MiniMax bid: $b_{mm} = c_2 = 180$
Total Bid: $b_1 = 0.5 \cdot 110 + (1 - 0.5) \cdot 180 = 145$

- **Robot 2**:
  Local task list:

$$LT_2 = [pos_5; pos_4; pos_9]$$

Cost starting from initial position pos_1:

$$c_1 = 20 + 20 + 30 = 70s$$

Optimal insertion of task T_{new}:

$$LT_2' = [pos_5; pos_4; pos_9; \mathbf{pos_14}]$$

New cost starting from initial position:

$$c_2 = 20 + 20 + 30 + 30 = 100s$$

Battery check:

$$\beta_2 = 45\% - (0.5 \cdot 100)\% = -5\%$$

This means that this robot cannot execute the extra task without charging. It needs to include an optimal charging station and an optimal charging time. The robot needs to charge at pos_2 and optimally includes this at a point the closest to its initial route, which is after visiting location pos_5. Its new local task list becomes:

$$LT_2'' = [pos_5; \mathbf{pos_2}; pos_4; pos_9; pos_14]$$

Going to the charging station takes $20 + 20 + 20 = 60s$ which results in a battery level of $45\% - (0.5 \cdot 60)\% = 15\%$. Going from the charging station to all of the other tasks takes another $20 + 20 + 20 + 30 + 30 = 120s$ and thus takes $(0.5 \cdot 120)\% = 60\%$ of battery which means that an extra of $60\% - 15\% = 45\%$ needs to be charged to fully complete all tasks, including the new one, without having charged more than needed. Looking at the charge model this would take an optimal charging time of $t_{opt} = 45\%/5 = 9s$. The total cost c_2 then becomes the sum of the total travelled time for executing LT_2'', which is $180s$, and the total charging time which is $9s$. Thus $c_2 = 189s$.
MiniSum bid: $b_{ms} = c_2 - c_1 = 189 - 70 = 119s$
MiniMax bid: $b_{mm} = c_2 = 189s$
Total Bid: $b_1 = 0.5 \cdot 119 + (1 - 0.5) \cdot 189 = 154s$

Both robots send their bids b_1 and b_2 to the auctioneer which follows the lowest bid principle:

$$b = min(b_1, b_2) = min(145, 154) = 145 = b_1$$

Despite that the new task clearly lies closer to the initial path of robot 2, robot 1 has the lowest bid and thus gets assigned task T_{new} due to the fact that robot 2 needs to charge when accepting the task. When no resource constraints would be implemented, robot 2 would have obtained the task, which would have resulted in a longer total execution time than needed. Both robots end up with their new local task lists:

$$LT_1 = [pos_7; pos_8; pos_13; pos_14]$$

$$LT_2 = [pos_5; pos_4; pos_9]$$

This process repeats for every task that needs to be as-

signed. The parameter ε can be tuned depending on the situation. In the following section, our approach is validated in simulation and is benchmarked to the industrial state-of-the-art solution.

5. Validation and benchmarking

In this section, we validate our approach and benchmark its performance to the optimization-based solution which represents the industrial state-of-the-art solution:

- For the task allocation, we assume that industry uses optimization-based heuristics to solve their task allocation problem. Some discussion with industrial AGV manufacturers confirmed their use of heuristics. In reality, each AGV manufacturer may have a different way of optimizing, but in this paper we arbitrarily took a genetic optimization solver as a benchmark because this is a widely used heuristic solver in combinatorial optimization problems.
- For the resource management, we assume that industry handles what the authors call 'threshold-100 charging' in which an AGV heads to the closest charging station when a threshold is reached and charges fully. Again, discussions with industrial AGV manufacturers confirmed this way of handling resource management.

We use the same layout as in fig. 1. We spawn all tasks at a random location and perform a task allocation for each approach considering all tasks together without any time dependencies. The robots always start at their depot station. We did two different experiments, one without considering resource constraints to show the performance of the task allocation approach independently, and one with considering resource constraints to show the results of an optimal charging behaviour. In both experiments and in both optimization approaches, the optimization objective is the minimization of the linear combination of the total travel cost and the total execution time. In the simulations, the tuneable objective parameter ε is set to zero, thus minimizing the total execution time of executing all tasks.

5.1. Without resource constraints

For this experiment, we do not consider resource constraints and assume that all robots have enough resources to finish their task sequences allocated by the task allocation approach. Our approach is implemented as stated in Section 3 but without the optimal charging inclusion in the bid calculation. For the industrial benchmark we use a genetic algorithm with a population size that equals ten times the number of robots, a mutation probability of 0.5, an elite size that equals the number of robots, and a number of iterations that equals ten times the number of possible solution combinations but which was limited at 1000 iterations. All these setup parameters are chosen arbitrarily to obtain clear results. We also calculate the optimal task allocation solution as an extra benchmark. We simulate for one up to four robots and this for one up to ten tasks. For each combination we simulate five times to analyse eventual stochastic variations. Fig. 2 shows an example assignment of the SSI-approach for ten randomly spawned tasks (coloured stars) and three robots (green, blue, and yellow squares). Fig. 3 shows the optimal assignment for the exactly same problem. The thick coloured lines (green, blue, and yellow) denote the path the robot follows. We can see that both approaches output a different solution with a different performance that is analysed further. Fig. 4 depicts the objective values for both the optimal, the heuristic (industrial benchmark), and SSI-approach for a series of assignments in function of the number of tasks and for one up to four robots. Remark that these objective values equal the total execution time of all tasks because the tune-able objective parameter ε was set to zero.

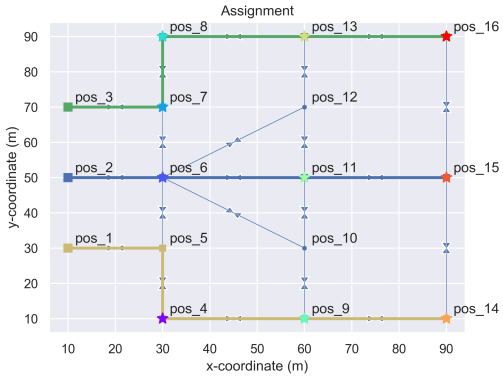


Fig. 2: SSI-auction assignment for ten tasks and three robots

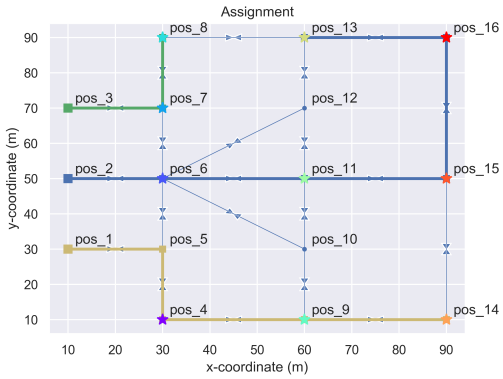


Fig. 3: Optimal assignment for ten tasks and three robots

Also remark that we did not include the optimal solutions for the situation of 4 robots. This because the computation time for a brute force allocation of eight, nine, or ten tasks is enormous. So we only showed the optimal results for one up to three robots which can easily be generalized to more robots. It can be seen that the SSI-approach outputs a performance in between that of the optimal solution and the heuristic solution for one to four robots, but this with not much difference. In the case of one robot, all performances are obviously equal because only one solution is possible, which is the single robot executing all tasks. For more than four robots, the performance stays roughly the same because there are enough robots for the job and more robots will not cause an increase in performance anymore.

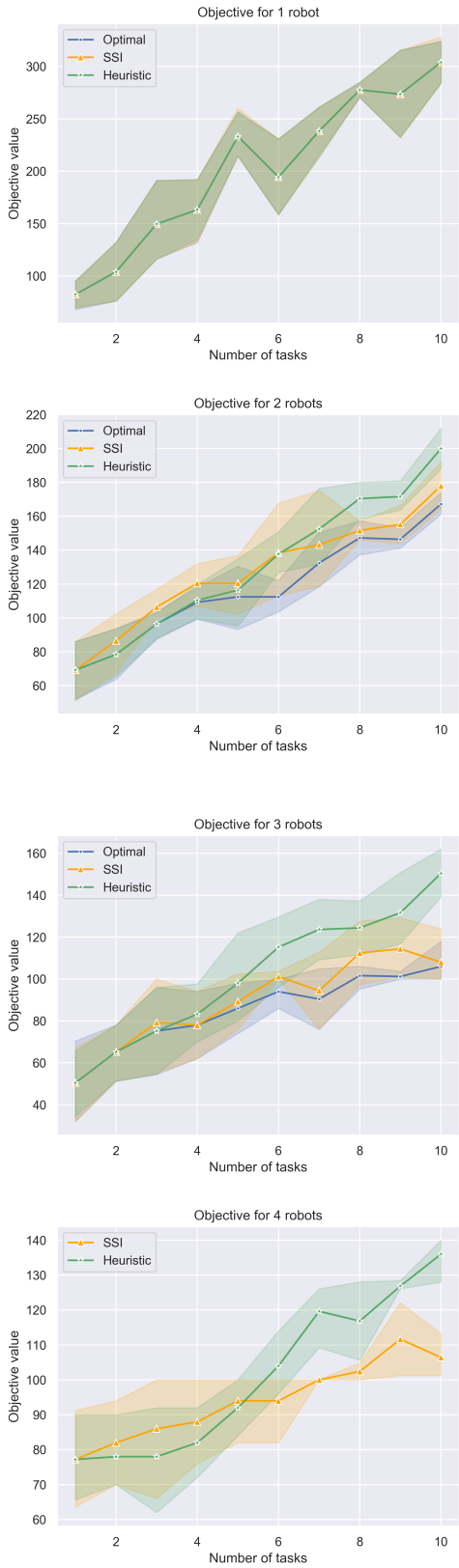
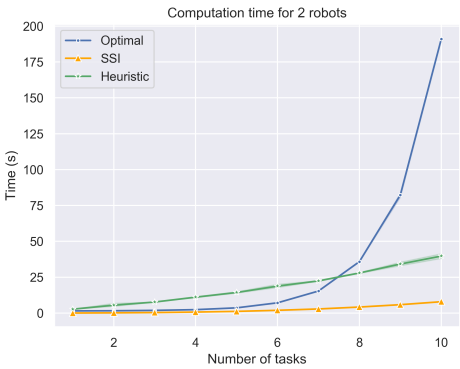
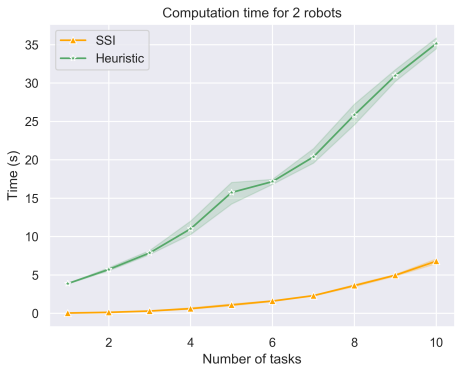


Fig. 4: Validation of the SSI-approach without resource constraints for one till four robots and one up to ten tasks.

Fig. 5 shows the computational time in function of the number of tasks for two robots. Fig. a shows this for the optimal, the heuristic (industrial benchmark), and the SSI-approach. Fig. b shows the same results but without the optimal approach to better compare the heuristic and SSI-approach.



(a) Computation time compared for the optimal, heuristic, and SSI-approach.



(b) Computation time compared for the heuristic and the SSI-approach.

Fig. 5: Computation time in function of the number of tasks for two robots and for one up to ten tasks.

We can see that the computation time of the optimal approach increases exponentially with the amount of tasks. We see that the total computation time for the SSI-approach is far less than for the optimal brute force algorithm and scales more linearly with the amount of tasks than the heuristic solution. We can conclude that the SSI-approach outperforms the industrial solution in computation time for equal performance when no resource constraints are considered. The computation time of the heuristic solution could be decreased by limiting the amount of iterations or limiting the population size. However, this would lead to a decrease in performance as the exploration of the search space would be limited. The heuristic approach thus asks for a trade-off between optimality and time complexity whereas the SSI-approach has both good performance and relatively little computation time.

5.2. With resource constraints

For this experiment, we do consider resource constraints and assume that all robots have a limited amount of resources and could need to charge in order to finish their task sequences allocated by the task allocation solution. In this experiment we compare our optimized resource management strategy from previous work and the industrial threshold-100 charging by including their outcomes in the task allocation approaches of the one proposed in this paper and of the industrial one, which is represented by a genetic algorithm, respectively.

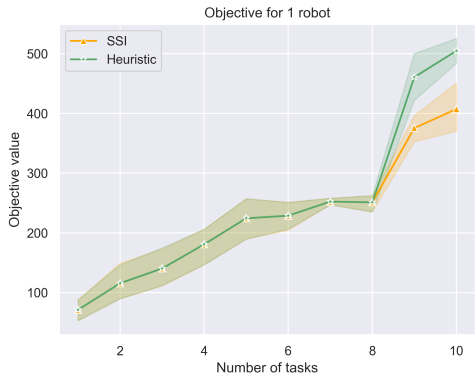
The setup for this experiment is the same as for the experiment without resources except for the implementation of a charging level for each AGV which starts at 100% at the start of the simulation, and the implementation of a resource consumption and a charging characteristic:

- For the resource consumption, we arbitrarily choose the function $deltaR_c(t) = -0.3 \cdot t$ to obtain clear results and to make sure that each AGV can reach a charging station from any location in the graph. In reality, this consumption function is more complex and depends on the loading, unloading, speed, and accelerations of the vehicle. However, this simpler function leads to the same conclusions about the performance of the approach.
- For the charging characteristic, we arbitrarily choose the function $deltaR_{ch}(t) = t$ to obtain clear results. Also in reality, the charging characteristics are more complex and non-linear.

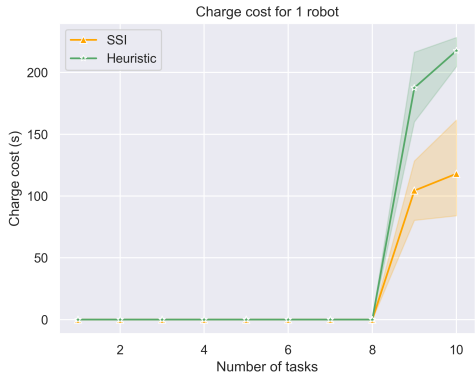
The solution approaches for this experiment are the same as for the experiment without resources except for the implementation of a charging scheme in both the industrial and our approach:

- For the industrial (central) approach, the task allocation is done as in the previous experiment, with a genetic algorithm solver. Once the solution is obtained, the robot checks if it needs to charge and applies the threshold-100 charging scheme if it does. In the simulation, we took a threshold of 20% as this is a general minimum charging level of the broadly used lead-acid batteries in AGV vehicles.
- For our approach, the optimal insertion approach as described in [10] is used to calculate an optimal charging station and charging time and the cost of charging is included in the bidding mechanism as described in Section 3.2.2.

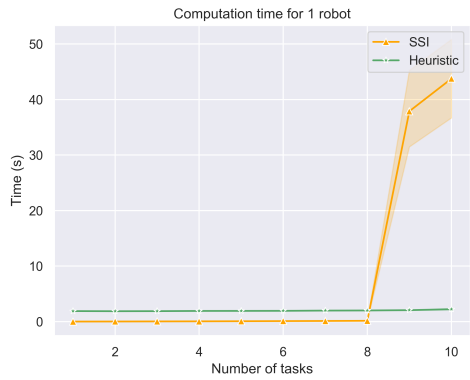
For both approaches, the total charging cost is calculated for each solution and is compared. This charging cost consists of the cost for moving from the initial tour to the chosen charging station, the cost to charge, and the cost to move from the chosen charging station to the next station in the task sequence. To clearly show the results of both resource management approaches independent of the task allocation results, only a situation with one robot is simulated for a set of tasks from one up to ten tasks and this again five times to verify the impact of stochastic variation. Fig. 6 shows the validation of the SSI-approach with charging constraints for one robot and one up to ten tasks. Fig. a shows the objective value of the SSI-assignments including resource constraints. Fig. b shows the charging costs of one robot in function of the amount of tasks for the industrial approach (Heuristic) and for the SSI-approach. We can see that our approach clearly needs less charging cost than the benchmark solution. The saved charging cost is due to the fact that the robot chooses the charging station the closest to its initial tour and decides to charge just as much as needed to finish its tour with the minimum amount of resources allowed. For situations with less than six tasks, the robot can finish its task sequence without charging, thus both approaches output zero charging costs.



(a) Objective in function of the number of spawned tasks for one robot.



(b) Charging cost in function of the number of spawned tasks for one robot.



(c) Computation time in function of the number of spawned tasks for one robot.

Fig. 6: Validation of the SSI-approach with resource constraints for one robot and one up to ten tasks.

Fig. c shows the computation time needed for each resource management approach in function of the amount of tasks. We see that our optimal approach needs more computation time due to the optimization processes, whereas in the benchmark solution, no optimization is used and just the closest charging station is obtained. To overcome this issue and to prevent the computation time of growing too much, the amount of tasks in each robot's task list could be limited as the time needed to obtain the optimal charging station into the sequence of tasks is proportional to the amount of tasks in the sequence. In this way, the total computation time for an optimal charging behaviour could be bounded and controlled. By implementing this optimal charging behaviour inside the bidding mechanism of a sequential single-item allocation process, it is possible to maintain a decentralized task allocation process under resource constraints. Both the task allocation and resource management tasks are handled separately and can be combined with only small adaptations facilitating their integration in existing industrial systems.

6. Conclusions

In this paper, we presented a decentral task allocation architecture for a fleet of AGVs based on the sequential single-item auction principle considering resource constraints. When comparing the task allocation approach without resource constraints to the industrial solution which is represented by a genetic algorithm solver, we clearly see that our solution outputs near-optimal performance with a computation time that scales linearly with the amount of tasks and amount of robots. This while the industrial reference solution scales much worse while outputting similar performance on the same task. When including resource constraints and comparing the charging costs of both our approach and the industrial benchmark, we see that our approach always wastes less charging time than the industrial benchmark but that it takes extra computation time to execute the extra resource optimizations. This could be limited by limiting the amount of tasks that can be assigned to one robot.

In conclusion, the proposed decentralized task allocation approach scales well with the complexity of the system and outperforms the industrial benchmark in computation time, for the task allocation, and performance, for the resource management. The task allocation and resource management task of the AGV can easily be combined by just adding the extra cost in the bidding procedure of the auction process. This easy implementation helps in a gradual adoption of decentralized control in multi-robot systems. In future research, we will also introduce routing information and uncertainty in the bidding process. In this paper we assumed path costs to be 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 modelled by using an imprecise uncertainty model [29, 30] which 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] K. F. E. Tsang, Y. Ni, C. F. R. Wong, L. Shi, A Novel Warehouse Multi-Robot Automation System with Semi-Complete and Computationally Efficient Path Planning and Adaptive Genetic Task Allocation Algorithms, in: 15th International Conference on Control, Automation, Robotics and Vision, ICARCV 2018, 2018.

[2] K. Jose, D. K. Pratihar, Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods, Robotics and Autonomous Systems 80 (2016) 34–42.

[3] C. Liu, A. Kroll, A centralized multi-robot task allocation for industrial plant inspection by using A* and genetic algorithms (2012).

[4] 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.

[5] M. P. Fanti, A. M. Mangini, G. Pedroncelli, W. Ukovich, A decentralized control strategy for the coordination of AGV systems, Control Engineering Practice 70 (September 2016) (2018) 86–97.

[6] D. Weyns, T. Holvoet, K. Schelfthout, J. Wielemans, Decentralized control of automatic guided vehicles : Applying multi-agent systems in practice, 23rd ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications,OOPSLA 2008, October 19, 2008 - October 23, 2008 (2008) 663–674.

[7] I. Baffo, G. Confessore, G. Stecca, A decentralized model for flow shop production with flexible transportation system, Journal of Manufacturing Systems 32 (2013) 68–77.

[8] G. Demesure, M. Defoort, A. Bekrar, D. Trentesaux, M. Djemai, Decentralized Motion Planning and Scheduling of AGVs in an FMS, IEEE Transactions on Industrial Informatics 14 (4) (2018) 1744–1752.

[9] 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.

[10] M. De Ryck, M. Versteyhe, K. Shariatmadar, Resource management in decentralized industrial Automated Guided Vehicle systems, Journal of Manufacturing Systems 54 (October 2019) (2020) 204–214.

[11] M. L. Gini, Multi-Robot Allocation of Tasks with Temporal and Ordering Constraints, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17), 2017.

[12] N. Sullivan, S. Grainger, B. Cazzolato, Sequential single-item auction improvements for heterogeneous multi-robot routing, Robotics and Autonomous Systems 115 (2019) 130–142.

[13] E. Nunes, M. Manner, H. Mitiche, M. Gini, A taxonomy for task allocation problems with temporal and ordering constraints, Robotics and Autonomous Systems 90 (2017) 55–70.

[14] B. P. Gerkey, M. J. Mataric, A formal analysis and taxonomy of task allocation in multi-robot systems, International Journal of Robotics Research 23 (9) (2004) 939–954.

[15] A. Khamis, A. Hussein, A. Elmogy, Multi-Robot Task Allocation: A Review of the State-of-the-Art, Cooperative Robots and Sensor Networks 2 (2015) 31–51.

[16] H. Eimaraghy, I. Manufacturing, S. Ims, Scheduling of Manufacturing Systems Under Dual-Resource Constraints Using Genetic Algorithms, Journal of Manufacturing Systems 19 (3) (2000) 186–201.

[17] Y. Fang, H. Xu, Q. Liu, D. Truong, Evolutionary optimization using epsilon method for resource-constrained multi-robotic disassembly line balancing, Journal of Manufacturing Systems 56 (June) (2020) 392–413.

[18] A. R. Mosteo, L. Montano, Simulated annealing for multi-robot hierarchical task allocation with flexible constraints and objective functions, IROS'06 workshop on Network Robot Systems: Toward intelligent robotic systems integrated with environments (2006) 1–8.

[19] X. Li, Z. Liu, F. Tan, Multi-Robot Task Allocation Based on Cloud Ant Colony Algorithm, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10637 LNCS (2017) 3–10.

[20] B. P. Gerkey, M. J. Matarić, Sold!: Auction methods for multirobot coordination, IEEE Transactions on Robotics and Automation 18 (5) (2002) 758–768.

[21] M. B. Dias, A. Stentz, A Free Market Architecture for Distributed Control of a Multirobot System, 6th International Conference on Intelligent Autonomous Systems IAS6 6 (2000) 115–122.

[22] R. G. Smith, The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, IEEE Transactions on Computers C-29 (12) (1980) 1104–1113.

[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, The power of sequential single-item auctions for agent coordination, Proceedings of the AAAI Conference on Artificial Intelligence (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.

[26] X. Jia, M. Q. Meng, A survey and analysis of task allocation algorithms in multi-robot systems, 2013 IEEE International Conference on Robotics and Biomimetics, ROBIO 2013 (2013) 2280–2285.

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

[28] R. Matai, S. Singh, M. Mittal, Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches, in: Traveling Salesman Problem, Theory and Applications, no. January 2014, 2010.

[29] K. Shariatmadar, K. Driesen, M. De Ryck, F. Debrouwere, M. Versteyhe, Linear programming under ϵ-contamination uncertainty, in: International Conference Computational and

Mathematical Methods in Science and Engineering, 2019.

[30] K. Shariatmadar, M. Versteyhe, Linear programming under p-box uncertainty model, Machines (2019).