# Automated Guided Vehicle Systems, State-Of-The-Art Control Algorithms and Techniques

M. De Ryck[a,*], M. Versteyhe[a], F. Debrouwere[a]

[a] *Faculty of Engineering Technology, KU Leuven, Spoorwegstraat 12, 8200 Bruges, Belgium*

**Abstract**

Automated Guided Vehicles (AGVs) form a large and important part of the logistic transport systems in today's industry. They are used on a large scale, especially in Europe, for over a decade. Current employed AGV systems and current systems offered by global manufacturers almost all operate under a form of centralized control: one central controller controls the whole fleet of AGVs. The authors do see a trend towards decentralized systems where AGVs make individual decisions favoring flexibility, robustness, and scalability of transportation. Promoted by the paradigm shift of Industry 4.0 and future requirements, more research is conducted towards the decentralization of AGV-systems in academia while global leading manufacturers start to take an active interest. That said, this implementation seems still in infancy. Currently, literature is dominated by central as well as by decentral control techniques and algorithms. For researchers in the field and for AGV developers, it is hard to find structure in the growing amount of algorithms for various types of applications. This paper is, to this purpose, meant to provide a good overview of all AGV-related control algorithms and techniques. Not only those that were used in the early stages of AGVs, but also the algorithms and techniques used in the most recent AGV-systems, as well as the algorithms and techniques with high potential.

*Keywords:* `AGV-systems, State-Of-The-Art, Control Techniques, Centralized, Decentralized, Potential Future techniques`

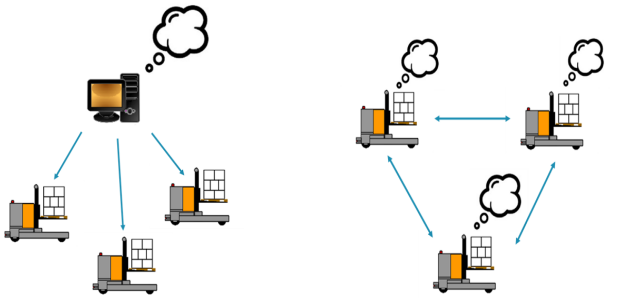
## 1. Introduction

Automated Guided Vehicles (AGVs) are mobile robots which are extensively used in the industry to transport goods from A to B. Currently, the market of AGVs is growing fast and is very dynamic. A market report published by Grand View Research (2017) [1] forecasting the period from 2018 to 2025 focuses on the potential growth opportunities of AGVs, stating that the future growth of AGV systems is (i) caused by the emergence of flexible manufacturing systems, (ii) the rising demand for customized AGVs and (iii) the adoption of industrial automation by SMEs. Current AGV-systems are well known and widely implemented in manufacturing, medicine, and logistics. In these systems, a fleet of AGVs is organized in a centralized way. Tasks like motion planning and allocation of tasks are done by a central entity, showed by Figure 1a, for all the AGVs together.

*Corresponding author

*Email addresses:* `matthias.deryck@kuleuven.be` (M. De Ryck), `mark.versteyhe@kuleuven.be` (M. Versteyhe), `frederik.debrouwere@kuleuven.be` (F. Debrouwere)

Driven by future requirements like flexibility, robustness, and scalability, the current trend in AGV systems is decentralization. The authors define decentralization as the distribution of the total intelligence of a system to its components: each device gets a part of the total intelligence to be able to operate independently, striving for the same global goals as depicted in Figure 1b.



(a) Central architecture (b) Decentral architecture

Figure 1: Central and decentral control architectures

The Grand View Research market report [1] states decentralization as one of the future technologies which will gain great attention. The most important reason why this trend is gaining attention, is the expansion of the AGV fleet. In future systems, larger and more complex systems will be needed to fulfill the transportation demand within a factory. This will not be feasible with currently employed systems because of memory, communication and computation limits. Academia and leading companies investigated decentralization resulting in rich publications which [2, 3] tried to review. This paper updates the state of the art and dedicates special attention to recent advances in practical applicable decentralization.

The authors decompose the AGV control into five distinct core tasks. Every core task is criticized in a decentral context. The remaining chapters are organized as follows. Section 2 starts with the Industry 4.0 context and how we see future AGVs fit in this context. In Section 3, a discussion on general decentralized control is provided as a prelude to decentralized control specific for AGVs. The main advantages and drawbacks of distributed control are discussed and the different paradigms to introduce decentralization in a system are described. In section 4, the core tasks needed to control a whole AGV-system are described. In the following sections 5 until 9, every core task is deepened out referencing the current existing state-of-the-art algorithms and techniques. To end each of these sections, a conclusion is made regarding which algorithms or techniques will be more prevalent in the future and thus, which will be more suitable for decentralized control of AGV systems. We complete the paper with a brief research discussion in Section 10 on how the AGV of the future looks like and how it will be controlled. Finally, we draw some general conclusions about every core AGV task in Section 11.

## 2. AGVs in an Industry 4.0 context

Industry 4.0 represents the fourth industrial revolution in manufacturing. The Industry 4.0 paradigm puts information central. The paradigm creates value from information extracted and refined from data. It is the paradigm by choice for our factories of the future which enable mass customization and allow further horizontal and vertical integration.

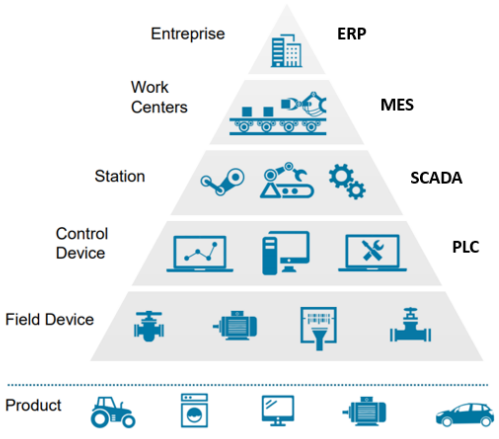



Figure 2: ISA 95 model [4]

With the possibility of free flow of data between elements in the production or in the logistics ecosystem, there is no need to rely on central architectures anymore to steer those

elements. As industry 4.0 models migrate from the typical automation pyramid ISA 95 to RAMI 4.0 (See Figure 2 and 3 respectively), members in a more complex system may communicate directly with each other and with their local environment on different levels. Intelligence can be distributed among the members and new architectures that generate value can be explored. Value here in terms of performance, scalability, robustness, and flexibility. In literature, the term 'Factory of the Future' is used as well to denote a factory reflecting these Industry 4.0 features. Also in the factory of the future, transportation is prevalent. With the use of mobile robots, an efficient and dynamic transportation of goods can be achieved. These mobile robots need to cope with the emerging requirements of Industry 4.0 as well. The fleet needs to adapt to changing circumstances, needs to be robust in any case, and needs to be scalable to any transport demand at any time.

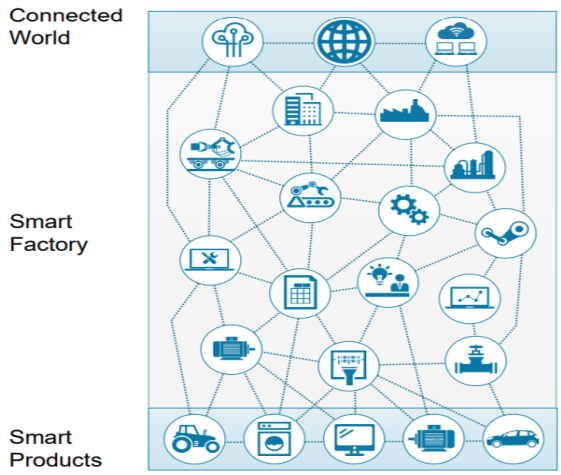



Figure 3: Axis 1 of the RAMI 4.0 model [4]

AGVs need to be intelligent, gathering useful information to make smart decisions very dynamically. Besides transport, the variety of tasks of AGVs will enlarge. In an Industry 4.0 context AGVs will not stay homogeneous. Using their intelligence and equipped with tools, they will have more functions other than transportation only. AGVs will be used more for "Ad Hoc" solutions. This in the sense that AGVs will be used for specific situations and will be equipped with tools to perform specific tasks. We expect a fleet of AGVs to be more heterogeneous, flexible, and dynamic where each vehicle will have specific abilities and will be able to operate in a flexible manufacturing system in a "plug-and-produce"-way. To realize the potential of the Industry 4.0 paradigm, AGVs need a different control architecture leaning next to big data, inter-connectivity, and cloud computing, on decentralization. The total intelligence of a system will not be centered anymore in one control unit but all devices will have their own intelligence creating data for independent information retrieval. This decentralization, and especially the adoption in general AGV control, is the specific aspect the authors are interested in. In the next section, this adoption in general manufacturing systems is discussed.

## 3. Discussion on the adoption of a decentralized control architecture

Decentralized control is one of the main features of Industry 4.0 paradigm. Many research is already conducted towards decentralized algorithms and techniques to control manufacturing systems in a distributed way [5–7]. Some research is done to the benefits of this architecture comparing to currently central and hierarchical structures. Many researchers mention the future need for decentralization [5, 8–17] recognizing the limits of the current central architectures as not suitable to handle flexible manufacturing, custom products, and complicated product specifications. [18] makes a comparison between current central and decentral control architectures in manufacturing and clarifies that centralized and hierarchical architectures are not compatible with the needs of future systems and that decentralization is the likely strategy to cope with the modern conditions. They state that decentral control fits for dynamic environments as it quickly adapts to changes. However, they also state the limitations of such a decentralized architecture. The main drawback in decentralized control is the increased effort needed to coordinate

all those independent entities as each of them tries to reach their own goals. This will not necessarily lead to the global optimum of the overall system. When adopting decentralized control architectures in a manufacturing process, there will always be a trade-off between optimality and flexibility. For small systems, decentralized approaches will be not as optimal as a centralized architecture but can guarantee more robustness and flexibility. However, because of the limitations of a central architecture, this will be, for larger systems, also be far from optimal. And this while the decentralized approaches can still guarantee robustness and flexibility even for very large systems. Table 1 compares both approaches.

| Centralized Approaches | Decentralized Approaches |
|---|---|
| Deeply rooted into the industry | Hardly implemented in industry |
| Well-known algorithms | Well-known algorithms |
| Access to global information | Access to local information |
| Global optimum | Sub-optimal |
| Small scaled systems | Large scaled systems |
| Simple systems | Complex systems |
| Not robust in dynamic situations | Robust in dynamic situations |

Table 1: Centralized vs. decentralized architecture

Different approaches to incorporate decentralization in a control architecture are developed in the last years. [5, 19, 20] gives an overview of various novel organizational principles, structures, and methods that can support cooperative behavior in future manufacturing. New approaches like holonic, fractal, random, biological and multi-agent manufacturing systems attempt to introduce a more decentral control architecture in future manufacturing systems. [16] introduces 'Anarchic Manufacturing' which incorporates the distributed control philosophy. This is an extreme form of decentralization in which the decision-making authority and autonomy is delegated to the lowest level of entities in system elements with no central control at all. These approaches are presented for general manufacturing purposes but can also be used in more specific areas like self-driving cars, unmanned areal vehicles, and of course automated guided vehicles.

**Remark**: This review paper has not as a purpose to review different decentralization approaches with their pro's and cons. The authors see general distribution of intelligence and computation as a way to overcome the limits of central systems and to cope with the requirements of Industry 4.0. So our purpose is to review the suitability of control algorithms to be used in general decentralized control architectures.

In the next sections, an overview of different algorithms and techniques used to control AGVs will be reviewed in the light of decentralized control architectures.

## 4. Core AGV Tasks

A complete AGV-system consists of multiple core tasks to be able to operate in complex environments. If an e-commerce warehouse is taken as a running example, then these core tasks can be easily distinguished. In such an e-commerce situation, there is a continuous stream of orders which enters the logistical warehouse system (ERP[1] or WMS[2]). An order can be seen as an object somewhere in the warehouse that needs to be brought to the picking station where the order can be sent to the customer. In this paper, the authors divide the total AGV-system in five core AGV-tasks shown by Figure 4.

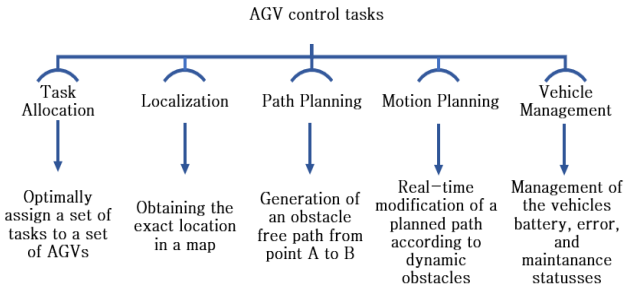



Figure 4: Overview of the five core AGV tasks

[1]Enterprise Resource Planning
[2]Warehouse Management System

A first core AGV task is Task Allocation. A set of tasks (orders) which has to be distributed to the fleet of AGVs need to be allocated to a specific AGV optimally. The easiest way to solve this is to allocate the task to the AGV which is closest to the position of the ordered object. Once a task is allocated, the next core AGV task is used to find the shortest path to the destination. This task is named Path Planning. It uses a representation of the environment to search for a sequence of segments to reach the goal as fast as possible. For Path Planning, it is important that the AGV can navigate properly in its environment. Thus Localization is also an important core AGV task. If a Path Planning algorithm computes the shortest path for an AGV, this does not mean that the AGV can follow that path without any problems. An unforeseen object or person can block the path or other AGVs may need some segments of the path at the same time. To avoid collisions or situations where multiple AGVs enter a life- or deadlock situation, there is another core AGV task named Motion Planning. This planner tries to avoid collisions with other static or dynamic objects. It tries to avoid deadlock situations and tries to limit the number of vehicles in a particular area. Limiting the numbers of vehicles in an area is called zone control. Once the collision-free path is executed and the AGV reaches its destination, the object can be loaded on the AGV. The exact same tasks are then used to bring the loaded object to the picking station. Parallel with all these core AGV tasks, there is another core task, Vehicle Management, which controls and monitors the status of an AGV. Some management issues are battery lifetime, maintenance requirements, and error status handling. A schematic overview of this generic AGV workflow can be seen in Figure 5.

In the following sections, every core AGV task will be treated and relevant algorithms and techniques will be described. To end each of the sections, a conclusion is made regarding which algorithms or techniques will be more prevalent in the future and thus, which will be more suitable for decentralized control of AGV systems

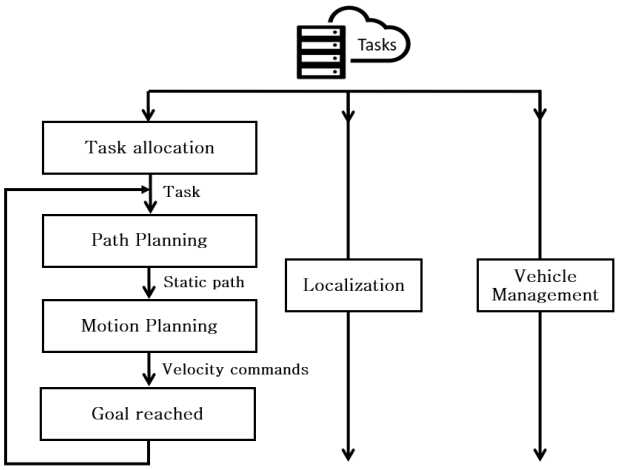



Figure 5: Threads of a generic AGV work flow

## 5. Task Allocation

Task allocation is one of the most challenging AGV tasks. It assigns a set of tasks to a set of robots, which can be seen in Figure 6.

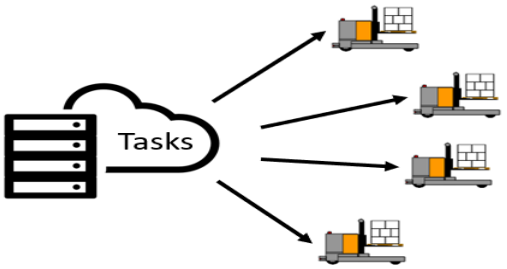



Figure 6: Task Allocation

This is a constrained optimization NP-hard problem [21] in which the cost of the total assignment needs to be as low as possible, which means that if there are a lot of tasks and robots, the number of solutions will be enormous. So there does not exist an efficient algorithm which can produce an exact solution to the problem in a finite amount of time. To solve these NP-hard problems, approximate optimization techniques named heuristics [22] and meta-heuristics [23] are used. A task in task allocation can be a particular object that has to be picked up at a certain location and

has to be dropped off at another location in a factory. Another task can be a surveillance task in which a group of AGVs must scan an area in the most efficient way. In the following sections, some properties and solution methods are described.

### *5.1. Desired properties*

In task allocation for AGV-systems, the most important properties [24] are (i) divisibility: the total work must be divided efficiently. The purpose is to have a high usage of every resource. Busy AGVs on one side and idle AGVs on the other at the same time is undesirable. And (ii) fault tolerance: the task allocation must operate correctly no matter what failures are faced.
Further properties are scalability, flexibility, and responsiveness.

- Scalability is the way the system can be enlarged without problems. The task allocation algorithms must keep working with a bigger amount of AGVs without reaching memory or computation limits.
- Flexibility means that the algorithm should, in any case, continue operating by continuously adapting to changes in the system.
- Task allocation should also be responsive, this means that it must have high performance also in dynamic environments.

To take all these desired properties into account, proper algorithms are developed. Section 5.5 covers these algorithms.

### *5.2. Taxonomy of tasks*

Every application requires another kind of task allocation. Gerkey and Mataric [25] have proposed a widely accepted taxonomy for task allocation in multi-robot systems. They divide the tasks into the following categories:

- Single robot tasks (SR): Tasks which only need one robot to be completed.
- Multiple robot tasks (MR): Task which needs more than one robot to be completed.
- Single task robots (ST): Robot which can only perform one single task at a time.
- Multi-task robots (MT): Robots which can perform more tasks simultaneously.
- Instantaneous assignment (IA): Tasks are independent of each other and there is no planning for future allocations. The available information about the task only permits an instantaneous allocation.
- Time-Extended assignment (TA): Tasks are dependent on each other. Future allocations can be planned considering several constraints. See section 5.3 for some dependency constraints.

A task allocation situation can be described by a triplet of these categories. The most simple situation is the triplet: SR-ST-IA. In which simple independent tasks, only requiring a single robot, are executed by robots which can only perform one task at a time. An extra division in task allocation problems can be made between static and dynamic allocation:

- Static task allocation: The tasks which are allocated are completed by the robot to which the task was initially allocated. Tasks cannot be re-assigned.
- Dynamic task allocation: The tasks can be re-assigned if there is another robot which is better to suit the task than the robot which was initially assigned to the task.

Many types of tasks can occur. For this reason, a bunch of algorithms is developed in past decades to meet all these different kinds of tasks covered in Section 5.5.

### *5.3. Task Constraints*

In some applications, tasks are independent of each other and are assigned to a robot from the moment the

task is available. The only relevant information about the task is a starting and an ending point. After the assignment, the robot can directly execute the task knowing this information. This can be the case in a warehouse environment where an order enters and the task for the AGV is to get the ordered object and bring it to the picking station. Knowing the basic information, the task can be assigned to the robot which is closest to the task. This is the simplest case. Real-world applications though set several constraints [24, 26]:

- Temporal Constraints: Tasks can have a time window in which tasks can have a duration, minimum starting time, and maximum ending time (deadline). These are time constraints related to the specific task.
- Precedence Constraints: There are also constraints which cause that tasks are dependent on each other. Tasks can be partial ordered, which means that some tasks must be completed before or after another task. Tasks can be coupled, which means that two or more tasks must be executed at the same time. There can also be incompatibility, in which tasks produce or obsolete other tasks.
- Some further kind of constraints which can restrict some assignments are mobility interferences. For instance when a narrow aisle exists where only one robot can pass on the way to the task.
- A last type of constraints is resource constraints. These can prevent a task to be executed when resources are empty. The AGV then has to be charged before it can execute more tasks.

For the expression of all these types of constraints, representations like Simple Temporal Networks (STNs) [27] or Hierarchical Task Networks (HTNs) [28] can be used. The presence of this variety of constraints has as a result that task allocation in multi-robot systems can get quite complex.

### *5.4. Optimization Objectives*

In the allocation of robots to a set of tasks, there is always a certain objective to be optimized [24]. There are several optimization objectives which can be used.
First, there are several elements which can be optimized:

- Cost: Cost that it takes for a robot to execute a task. This can be travel cost like time, distance, or fuel consumption.
- Fitness: How well a robot can perform a task.
- Reward: Gain of completing a task.
- Priority: Urgency of completing a task.
- Utility: The subtraction of cost from reward or fitness.

Second, there are some types of objectives possible:

- MinMax: Minimize the cost of the worst robot.
- Egalitarian: Maximize the utility of the worst robot.
- TotalSum: Minimize the sum of individual costs.
- Maximize the sum of individual utilities.
- Minimize the average cost per task.
- Maximize throughput.

Different solution approaches use different optimization objectives to get to the global optimum. These solution approaches will be discussed in the next section.

### *5.5. Solution Models*

In the last decade, there has been done a lot of research to solve the problem of multi-robot task allocation. It is a vast area because of the huge diversity of tasks and task constraints. A lot of solution approaches have been developed and are available in literature. This

makes structuring of task allocation algorithms rather difficult. In this review paper, the authors make two clear separations in the state-of-the-art solution algorithms: (i) optimization-based solutions and (ii) market-based solutions. Each of these two solution methods are covered in the next sections. The other two, behavior-based, and field-based methods, have limited use and are only briefly described for completeness. Figure 7 gives an overview.

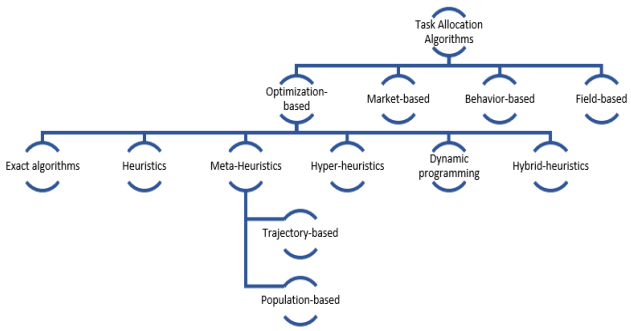



Figure 7: Overview Task Allocation algorithms

As mentioned in Section 3, the whole AGV control can be architectured in a more central or a more decentralized way. In the following, the authors will describe the four core task allocation solution approaches. Each approach will have a less or more decentralized character which will mark the approach less or more suitable in a decentral context

### *5.5.1. Optimization-based solutions*

In optimization-based solutions [29], an algorithm searches for an optimal solution in a solution space which maximizes a profit or minimizes a cost using global information and considering all constraints. This is visualized in Figure 8. If the solution space is small, exact solutions can be found. For a small number of tasks and robots, it is, for example, feasible to generate a matrix with traveling costs for each robot to get to each task. Taking the constraints into account, it is possible to use an exact algorithm which can find the optimal solution of task assignment in this small solution space in a finite amount of time. If the amount of tasks and robots increases significantly, the solution space becomes too large. In this situation, no exact solution methods can be used. These problems are called NP-hard and no exact algorithms exist which can find the optimal solution in a finite amount of time and approximate search algorithms like heuristics, meta-heuristics, or hyper-heuristics need to be used.

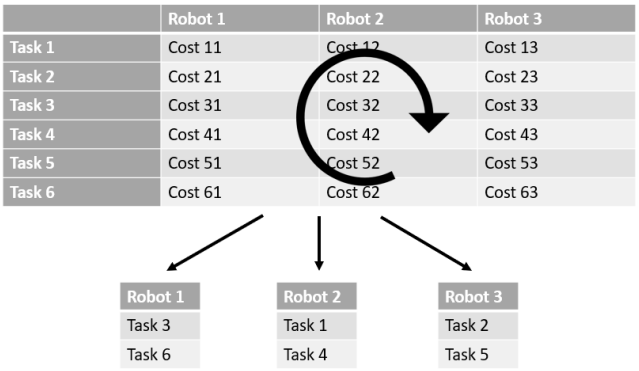



Figure 8: Optimization principle

*Exact algorithms.* A first exact searching algorithm is Mixed-Integer Linear Programming (MILP) [30], where the objective functions and constraints are formulated in integer and linear equations and solved by a certain solver algorithm. Another exact algorithm is Brute Force Search or Exhaustive Search [31]. This algorithm evaluates every possible solution and selects the best. The Branch and Bound algorithm [32, 33] is also an exact algorithm. In this algorithm, the set of possible solutions is represented as a tree with the whole set of solutions at the root. The algorithm explores branches of this tree which represents subsets of solutions. For each branch, an upper and lower bound is defined. The algorithm enumerates candidate solutions of a branch if this branch can produce better solutions than the solution already found. Otherwise, the branch is discarded and no solutions in this branch will be evaluated. Also, other tree-based algorithms [34, 35] can be used to find an exact solution. As stated earlier, these algorithms can only be used for small solution spaces. This means a very small amount of robots and a small number of tasks to allocate. If there are $n$ robots and $p$ tasks

to allocate, then there are $n * p$ possible allocations. This number can increase very quickly, making exact algorithms of little use for normal sized- and large-sized systems. And certainly if also complex constraints are considered.

*Heuristics.* Heuristics like Random Search [36], or Hill-Climbing [37] take a solution every time step during a fixed amount of steps and compare it to the best solution already found. The best solution yet found after the amount of steps to execute is considered as the best solution. A more recent heuristic approach is the nCAR algorithm [38]. This nearest-neighbor based Clustering And Routing algorithm is based on the known Vehicle Routing Problem (VRP).

*Meta-heuristics.* Heuristics only look for better solutions when comparing. Using this technique, they can get stuck in local optima. Therefore, meta-heuristics are used which also temporarily allow worse solutions at some times to get out of the local minimum. These are actually strategies to guide a search process. Meta-heuristics can be divided into trajectory-based and population-based methods:

- Trajectory-based methods are methods where a solution space is searched and where the probability of choosing a better solution above a worse one is dependent on the moment of the trajectory in the timespan of the algorithm. Examples are Simulated Annealing [36, 39, 40], Iterative Local Search [41], Variable Neighborhood Search [42], and Tabu Search [36].
- Population-based methods are methods where populations are used to search for the solution space. Examples are Genetic Algorithms [43–45], Particle Swarm Optimization [46–49], Memetic Algorithms [50], and Ant Colony Optimization [51].

*Hyper-heuristics.* Hyper-heuristics [52] are used to automate the process of selecting, combining, adapting, or generating several heuristics to solve search problems. There are a lot of heuristics which can be chosen to solve a problem. Each has their own weaknesses and strengths. Hyper-heuristics try to automatically choose proper heuristics out of a set of low-level heuristics at any given time dependent on the current state of the problem.

These heuristics, meta-heuristics, and hyper-heuristics can handle much larger solution spaces than exact algorithms can do. But for very large solution spaces, it is less likely to find the global optimum. The larger the solution space gets and the more constraints exist, the more difficult it gets to find this optimum.

*Dynamic Programming.* Dynamic programming [53] is another method as a method to solve problems in general which does not really fit into the above-mentioned structure. In dynamic programming, the total problem is divided into subproblems of which a solution can be found in a more simple way than the solution of the total problem. This is the case if the sub-problems can be recursively nested into the global problem. The total complex problem can be solved by using the solutions of the simpler sub-problems. Results of sub-problems are stored in a kind of table. When solving the total problem, these solutions can be used when needed. In optimization, dynamic programming simplifies a decision by breaking it down into simpler decisions, which is called 'Divide and Conquer'. Because of the division of the total problem, it seems that Dynamic Programming is better able to find a more optimal solution than heuristics do. But still, for very large fleets, it is hard to keep up good system performance.

*Conclusion.* In this section, the most widely used optimization-based task allocation algorithms were covered. These algorithms need to search a solution space for feasible solutions. This is a global way of optimizing. The algorithms need to have access to the global information to be able to search for the optimal solution. Optimization-based algorithms are widely used in centralized task allocation

algorithms and have great performance for small AGV systems. However, for real-life systems, the solution space just gets too big causing the need for heuristics which only find near-optimal solutions. And as the system gets larger, it gets more difficult to find a good solution which approximates the global solution. When computation gets heavier due to size and complexity, this also pushes the boundaries on the computer's performance, which is again a cause of performance decrease. A second issue is the lack of robustness.

### *5.5.2. Market-based solutions*

In market-based solutions [29], an economic principle is used to solve the task allocation problem. In this case, the allocation is not done by executing an optimization process using all available information. But a specific method of auctions is used where each robot uses its local information to calculate bids. The most basic approach of the principle of an auction is the CNET protocol [54]. In this protocol, an auctioneer announces one single task at a time to bidders. Each bidder places a bid for the task-dependent on the cost for the bidder to execute the task. The auctioneer then evaluates all the bids, including its own bid, and assigns the task to the robot with the highest bid. This is called a greedy solution as the simplest best solution is chosen. After a robot is assigned to a task, it can no longer bid on another task. The auction process is visualized in Figure 9.

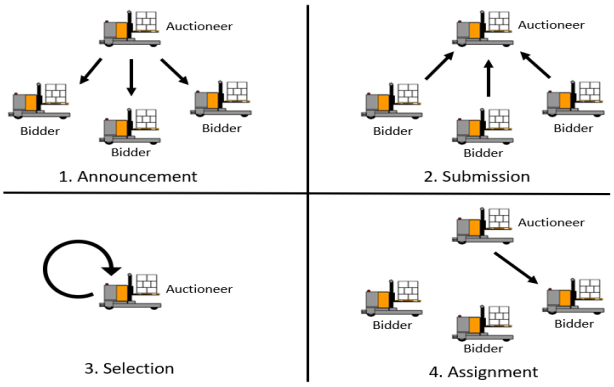



Figure 9: Auction principle

This is the simplest market-based approach but there are a lot of variants to this as reviewed in the following parts.

*Role of the auctioneer.* First of all, the role of the auctioneer can be played by a central computer, by a fixed robot, or can be altered between robots based on the winner, at random or via token passing [55]. This causes that a market-based approach can be central or decentralized from structure. The way of optimizing in all different methods stays the same if all robots cooperate in the same auction. This because the auction process stays the same. All robots compute their bids and send them to a decision-maker which chooses the best allocation based on all bids. At which robot or computer this decision maker is situated does not matter, the result of allocation will be the same. Although, looking at robustness and flexibility, the role of the auctioneer does have an effect. By using an altered role, the robustness of the system increases as there is no single point of failure. The task allocation process may be altered between robots, making it robust and suitable for decentralized control. Hence, market-based allocation using a central computer may facilitate later deployment to a full decentral control. This is also an important aspect to consider as it can be interesting for a company to adapt gradually the current central architecture. As a last remark, in really large fleets, it could also be possible to have multiple auctioneers in parallel: Only AGVs within a certain radius from the auctioneer, can bid on its tasks.

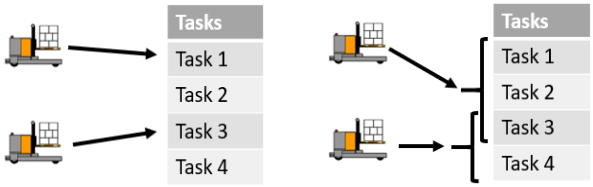
*Auction principle.* The way of offering tasks to robots can differ. There can be one single item auctioned at a time. This is called a sequential single-item auction and is used in most of the algorithms like CNET [54], SIT-MASR [56], CBAA [57], and OCA_Alloc [58]. Also, a bundle of items can be auctioned together. This is called a combined auction. In this last category, there are another two options:

- Parallel single-item auction: Robots can bid on a bundle of tasks but with one bid for each task. The

task which received the highest bid is then allocated to the robot which offered this highest bid and the unallocated tasks are auctioned again. This is applied in the Prim Allocation [59] algorithm. This can be seen in Figure 10a.

- Combinatorial auction: The robot can bid on a combination of tasks in which it has one bid for a cluster of tasks. This technique is used to prevent synergies. The CBBA [57] and SET-MASR [56] algorithms use this approach. This can be seen in Figure 10b.

**Synergy:** If there are two tasks and the cost for one robot to go to each task separately is higher than if it does the tasks in sequence (combination), then this is called a positive synergy between these tasks. This is the situation when both tasks are located near each other. A negative synergy between tasks means that the tasks are located far from each other and the cost for the robot to visit each task separately is smaller than to visit the tasks in sequence.




(a) Parallel single-item auction (b) Combinatorial auction

Figure 10: Auction principles

Combinatorial auctions are known to have solutions near to the global optimum. But this has as a price that they are computationally heavy because of the number of possible task combinations which is exponential in the number of tasks. Sequential single-item auctions are mostly used because they are simple yet effective. [60]compares sequential single-item auctions, parallel single-item auction and combinatorial single-item auctions: They state that the coordination system based on sequential single-item auctions is more suitable than the others as it combines the advantages of combinatorial auctions as well as parallel single-item auctions. They proved that the implementation of the coordination system based on sequential single-item auctions results in no more bids than parallel single-item auctions. On top of that, they notice that it is much easier to implement than combinatorial auctions since the central auctioneer receives exponentially less information and does not need to solve an NP-hard problem and that it provides much better performance guarantees than the coordination system based on parallel single-item auctions. [61], the usefulness of sequential single-item auctions is emphasized. [62] presents a distributed algorithm compared to the widely used sequential single-item and combinatorial auction methods. They conclude that their algorithm can find the global solution in contrast to sequential single-item and combinatorial auction algorithms which can only find local optimal solutions. However, they also concluded that the computation complexity increases dramatically as the scale of robots in the system and tasks grows. And thus their algorithm only finds the global solution for small and simple systems, which in the light of future systems, makes the algorithm not relevant. As a result, the authors believe that sequential-single-item auctions will be the most suitable market-based algorithms for future task allocation: they are simple, yet very effective and can consider synergies. Although they find solutions further away from the global optimum than the methods following the combinatorial principle, they are not that computationally heavy and may consider more complex constraints. In the next sections, methods will be described which can increase the performance of these single-item auctions which definitely makes it a good consideration for task allocation in future systems.

*Participation rules.* The way robots are allowed in the participation in the auction can alter. There can be a method where a robot which won a bid is not allowed to the auction anymore. Or it can be possible that no matter how many tasks a robot is already assigned to, it can still

bid on tasks and participate in the auction. Looking at flexibility, it will be important in the future that an AGV can accept more tasks at once. Accepting task by task, only accepting a task after executing the previous one is far from flexible. This way of participating will provide the AGV with a local list of tasks it needs to execute. This makes it possible for AGVs to locally switch tasks with neighboring AGVs or to reassign their tasks when having an error status or when they need to charge. Another possibility is that AGVs can optimize the sequence in which they execute the tasks. This as an extra local optimization beside the auctions to increase overall performance. Having these local smaller optimizations, heuristics or even exact algorithms can be used.

*Bid calculations.* The way bids are calculated is very important. Bids can be calculated only using the cost for the robot to perform the specific task. This simple bidding is used in CNET, OCA_Alloc [58], CBAA [57], and CBBA [57]. But bids can also be calculated using the marginal (extra) cost for the robot to perform the task considering other tasks in its task list. This is used in the Prim Allocation [59], SIT- and SET-MASR algorithms [56].

**Marginal cost**: If there are two tasks of which task one lays on a distance of 5 m from the robot and task two on a distance of 10 m from the robot and on a distance of 3 m from task one, see Figure 11.

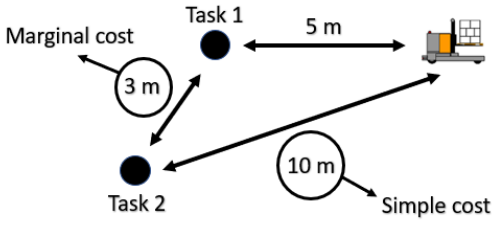



Figure 11: Marginal cost calculation

The total cost for the robot to execute each task separate is 15 m (5+10). But if the robot is already assigned to task one (cost 5 m), then after executing this task, it only takes the extra cost of 3 m to execute task 2. This extra cost is the marginal cost which is also used to prevent synergies. In this situation, there is a positive synergy between the two tasks.

To have a good performance on task allocation, the calculation of bids based on the marginal cost is the most beneficial. In the previous section, the authors stated that it is beneficial that AGVs accept more tasks at once to have higher flexibility. This causes them to have a local list of tasks they are supposed to execute. As AGVs will have this list, which can be locally optimized, it is important to also consider them in the calculation of a bid on a next task. The possibility exists that an AGV computes the lowest extra cost of inserting this new task into the already present task sequence. It can then bid this extra (marginal) cost which is more representative than bidding the distance to the new task.

*Constraints in market-based approaches.* Market-based approaches also have to deal with constraints. Variations on the standard mentioned algorithms exist which take these constraints into account. A first variation on the sequential single-item auction algorithm is the TeSSI algorithm [63]. A simple single-item auction where robots can bid on more tasks at the same time is used. But while bidding, each robot takes time constraints into account using a Simple Temporal Network (STN). This network is an individual schedule for each robot which uses it to find a free place in this schedule for a new task before it bids on it. Another distributed algorithm considering task deadline constraints is presented in [64]. A second variation on the sequential single-item approach takes precedence constraints into account [65]. And a third variation combines the previous two and considers both temporal and precedence constraints [66]. A last variant on the sequential single-item approach considers resource constraints [67].

The strength of market-based methods is that the total computation is distributed. This means that more complex computations on each device are possible without pushing computational limits. Thus it is possible to add many constraints when computing bids on tasks or when allocating the tasks. This may not be possible when working in a centralized architecture: more constraints mean more effort for the central unit to do the optimization, and thus a likely chance not finding the optimal solution as central controllers are supposed to do. The authors do see extra motivation here to move towards decentralized control in very large and complex systems handling lots of constraints.

*Additional consensus phase.* Some market-based algorithms also add an extra consensus phase to the auction process to better the quality of the assignments. This quality can be poor when only using auctions as a task allocation algorithm. A consensus in this sense is a further transfer of tasks between robots after the tasks are assigned by the auction. So there is a constant re-assignment of tasks during the operation of the AGVs. The above mentioned SIT-MASR [56] algorithm also uses consensus by exchanging single tasks after the auction. A variant on this algorithm is called SET-MASR [56]. In this algorithm, the negotiation is done with a set of subtasks instead of single tasks to improve the quality of solutions. An article which proves the advantages and increase in solution quality of task switching is [68]. Algorithms which also use consensus are CBAA, CBBA [57], DMB [69], OCA_Alloc [58], and some more [70, 71].

When distributing the intelligence, there is a migration from making global optimization's towards making more local optimization's. If AGVs will only local optimize without considering neighboring AGVs' intentions, the global solution will be high sub-optimal. Hence, it is very important in future AGV control, that AGVs mutually exchange information and try to strive for a global goal together. Adding this consensus phase to the auction process is thus, in the opinion of the authors, essentially in future task allocation. In [68], there is proven that this consensus phase definitely contributes to performance in market-based allocation. Thus this is required when tending towards a decentralized architecture. The authors see the sequential single-item auction in combination with a consensus algorithm as a very suitable task allocation combination in future large and complex AGV systems.

*Other market-based approaches.* Other market-based approaches are TraderBots [72], MURDOCH [73], and DynCNET [74]. Also a market-based approach for tasks which requires cooperation among robots is presented in [75]. A more recent auction-based task allocation algorithm [76], does not only consider cost as an objective to minimize but it also considers an even task distribution over a heterogeneous group of robots. [77] is a multi-agent-based approach which uses the multi-agent paradigm to effectuate the market-based approach for dynamic scheduling of AGVs in manufacturing systems.

*Conclusion.* In general, market-based approaches are robust and scalable and accept a lot of flexibility and complexity due to the distribution of computation. The optimization is done selfishly from the perspective of each AGV. But if every AGV tries to optimize itself, then also the global situation will be optimized. This will increase when AGVs cooperate with their neighbors and exchange tasks locally. Because of their decentralized nature, robustness, and the possibility to scale to large and very complex systems without pushing the boundaries, the authors see this market-based task allocation approaches the most suitable for future task allocation. Being aware that market-based approaches will never reach the global optimum as optimization-based algorithms do for small and normal-sized systems, the authors see especially see scalability, robustness, and flexibility of higher value than

reaching global optima. This is always the main trade-off which has to be made when opting for decentralized control.

#### *5.5.3. Behavior-based solutions*

In behavior-based solutions, robots use motivational behaviors such as impatience and acquiescence. Using these behaviors, robots can motivate the ability to perform a task or they can give up tasks they are not able to perform. Algorithms for this type of solutions are Alliance [78, 79] and Vacancy Chains [80]. These algorithms are seldom used in task allocation, we will not go further into them.

#### *5.5.4. Field-based solutions*

A very specific type of solution is the field-based solution [81]. It is not frequently used but can be added to another solution method to improve solutions. The essence of the algorithm is that a robot moves along a potential field which consists of attracting fields emitted by the goal and repelling fields which are emitted by obstacles and other AGVs. The superposition of those fields generates a total field of which the robot follows the gradient until it reaches the goal. As these algorithms are not very popular in task allocation, we will not go further into them.

### *5.6. Conclusion*

In this section, the authors went through a lot of different task allocation algorithms with different characteristics and solution methods. A lot of these algorithms are studied well and are implemented in practice. Nowadays, a lot of centralized approaches are implemented which make use of optimization-based algorithms which uses all possible information to make an optimal solution. But in the future, as many manufacturers may want to decentralize their systems and introduce flexibility, robustness, and scalability, more market-based solutions will be used. As already states earlier, the interest of the authors goes to the sequential single-item auction-based algorithms which introduce consensus to have better quality assignments. This is a combination of all the advantages of other algorithms without being too complex. Because the computational effort of these algorithms is not that high, it is possible to consider complex constraints in the optimization. By altering the auctioneer, a very robust system without a single point of failure can be created. AGVs will have a local task list which they can locally optimize using optimization-based algorithms which have high performance on these small-scaled optimizations. AGVs can use this optimized local task list to calculate the extra cost of inserting a new task in the most optimal position in the sequence. As an addition, tasks can be interchanged between neighboring AGVs which further improves performance. Table 2 gives an overview table of all task allocation algorithms. The table mentions the properties of each algorithm and shows the suitability of using them in a decentralized control architecture for industrial AGVs. Also here, Behavior- and Field-based approaches are not mentioned because of the little use in industrial AGV systems.

## 6. Localization

As for any vehicle moving inside, also in AGV control localization is one of the core tasks to consider. In contrast to the other tasks, this task is mostly already decentralized: all the localization equipment and software are onboard. Information about the location in the 2D-map of the environment can be communicated to a central computer or to neighboring devices for other control purposes. As this core task is independent on the control architecture, we will not discuss the suitability of each of the localization options for a particular control architecture. But for completeness of the paper, we will review the existing localization methods and criticize them regarding flexibility and robustness.

In the past decades, there has been a lot of improvement in localization systems [82, 83]. There are some old

| Task Allocation Algorithms | | |
|---|---|---|
| Algorithm | Advantages | Disadvantages |
| Optimization-based | | |
| Exact algorithms | Finds optimal solution for very small fleets | Only usable for very small fleets |
| Heuristics | Finds approximate solution for small fleets | Lack performance in large and complex fleets, |
| Meta-heuristics | Finds approximate solution for medium-sized fleets | Highly computational and time expensive, |
| Hyper-heuristics | Finds approximate solution for medium-sized fleets | Not flexible, not robust, not scalable |
| Dynamic programming | Finds approximate solution for medium-sized fleets | Not flexible, not robust, not scalable |
| Market-based | | |
| Central auctioneer | Good bridge between central and decentral architectures | Single point of failure |
| Floating auctioneer | No single point of failure (robust) | More complex algorithm to change auctioneer |
| Sequential single-item auction | Simple algorithm, introduces flexibility and scalability | High sub-optimal solutions |
| Parallel single-item auction | Simple algorithm, introduces flexibility and scalability | High sub-optimal solutions |
| Combinatorial auction | Near-optimal solutions, considers synergies | Computationally heavy |
| Restricted participation | Robot only needs to care about one task | Lacks flexibility, does not consider synergies |
| Non-restricted participation | Robots can locally optimize a task list, consider synergies | Robot needs ot care about more tasks at once |
| Simple bid calculation | Simple calculation of bids | Bid calculations does not represent the real costs |
| Marginal bid calculation | Does represent the real costs, considers synergies | None |
| Temporal constrained auctions | Problem description lies closer to the real-world problem | More complex computation |
| Precedence constrained auctions | Problem description lies closer to the real-world problem | More complex computation |
| Resource constrained auctions | Problem description lies closer to the real-world problem | More complex computation |
| Auction with consensus | Increases performance | More complex cooperation needed |

Table 2: Overview table Task Allocation algorithms

school techniques still in use and some newer techniques which will gain more attention in the future. Some of the older and proven techniques are:

- Inductive localization
- Optical localization
- Magnetic localization
- Inertial localization

Some new techniques which are rapidly gaining attention are:

- Laser localization
- GPS localization
- Natural localization
- Vision guided localization

Some localization methods make use of the physically present circuit to obtain a location. The authors gave these methods the name "physical path localization methods". With the term "circuit", we mean the layout of paths and intersections on which the AGV is supposed to drive. Other localization methods do not need a physically present circuit to localize themselves. The authors gave these methods the name "virtual path localization methods". Here, the predefined circuit can be maintained virtually. All the old localization methods are methods which use a physical circuit. For future requirements like flexibility, it is not efficient to have fixed physical paths which are difficult to adapt. Future factories are dynamic and can change in configuration. Virtual localization is much more flexible as a change in the circuit can be done easily online in the graphical design software. The authors prefer this kind of localization for flexible AGVs in the future. Figure 12

gives an overview.

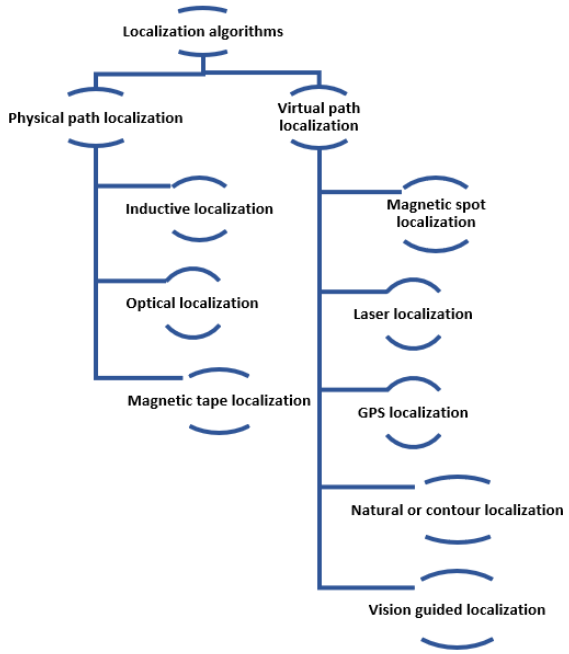



Figure 12: Overview Localization algorithms

### *6.1. Physical path localization*

With physical path localization, the paths are present as physical guidelines on the floor and the localization is straightforward with a single sensor detecting the guideline. The predefined paths are fixed on the floor using tape or embedded into the floor using a wire. The AGV does not actually know its position inside the area map but just stays on the track. Marks along the track [84] can tell the AGV whether it has to take a special action like increasing or decreasing speed or to rotate at a certain degree when being in a curve. In what follows, different techniques which use this physical path navigation are reviewed.

#### *6.1.1. Inductive localization*

This is the first type of localization used in the first generation of AGVs [85, 86]. In this type of localization, a wire is embedded into the floor running on electricity, generating magnetic flux. The AGV has a sensor onboard which consists of coils picking up the emitted magnetic flux so a controller may adapt the speed of the wheels. This is a proven technique especially in very small aisles to stay on track accurately but it isn't flexible.

#### *6.1.2. Optical localization*

In this type of localization, a color tape or a painted line with high contrast with the ground color is placed onto the floor [87]. The AGV has an optical sensor onboard and the localization principle operates similarly to inductive localization with the same benefits and disadvantages. Another disadvantage here is that the tape can become dirty or can be damaged. Yet it is easier to adapt than the above wire method. This option is also cheap because of the tape and the only use of an optical sensor.

#### *6.1.3. Magnetic tape localization*

The physical guide path is marked with magnetic tape that is placed onto the factory floor. Inside the vehicle, there is a magnetic sensor that can detect the magnetic field. The localization principle is similar to the above mentioned optical localization with the same benefits and disadvantages.

### *6.2. Virtual path localization*

With virtual path localization, the paths are virtually present inside the local map maintained by the AGV or in the global map of the central unit. This makes it easily adaptable and expandable. However, virtual localization is more difficult because the AGV needs to know its exact position into a 2D map. This in contrast to physical path localization where the AGV only needs to know its position on a 1D circuit. Knowing the location in the 2D map, deviations from the virtual path can be calculated.

#### *6.2.1. Magnetic spot localization*

A grid or line of spots is embedded into the floor [88] on specific $(x,y)$-coordinates in the area map. The spots can be passive permanent magnets or transponders. Inside the vehicle, there is a magnetic sensor that can detect

the spots. By detecting the spots, the AGV can determine its absolute position in the map. The location in between the spots is gathered using relative positions using encoders on the wheels which calculates the traveled distance (odometry). Due to the combination of absolute and relative localization, this is a very accurate method. A disadvantage is that this method is time-consuming to install and to modify.

#### *6.2.2. Laser localization*

Laser localization [89] is currently the most accepted method for AGV localization. A rotating laser is mounted onto the vehicle. For localization, multiple fixed reference points like reflective strips, are located in the operating area on known coordinates. The coordinates of the reflectors are added to the global map. The emitted laser beams are reflected and scanned by the AGV after which the AGV can triangulate its absolute position based on the coordinates of the reflectors. At least three landmarks have to be visible to be able to navigate. This is a very accurate, secure and reliable method and is now used as a standard in a lot of AGV-systems. Disadvantages are the high price and the effort to place all the reflectors in the factory area.

#### *6.2.3. GPS localization*

In GPS localization [87], satellites with known positions into the global map emit signals which are detected by the GPS-receiver. This receiver can then measure the distance to each satellite. This info is used to determine the absolute position of the receiver using trilateration. At least four satellites have to be visible to be able to navigate. For this technique, a clear line of sight to the sky is needed. This is difficult to obtain in industrial environments. As an alternative, a Local Positioning Radar (LPR) in the factory can be used instead of satellites. The disadvantage of this LPR is that there is a precision of 10 cm, which is not very accurate but may be improved using sensor fusion, see Section 6.3.

#### *6.2.4. Natural or contour localization*

This type of localization uses a Light Detection And Ranging (LiDAR) sensor to scan the whole environment around the vehicle [87]. No fixed landmarks like reflectors are needed. The AGV uses features in the existing environment to navigate. This makes this type of localization very flexible as no extra infrastructure is needed. Unfortunately, the method is not that precise and robust due to reflections and drift. Using the scanned map of the environment, the vehicle can make a 2D map of its surroundings with all visible features like walls and pillars. When comparing this local map with the map of the factory, the robot may infer its position into the map using Simultaneous Localization And Mapping (SLAM) [90]. Using SLAM, the robot explores the area while using laser scans for updating a local 2D-map causing the system to become a lot more flexible in dynamic environments. The disadvantages are that the sensors are expensive and that some transparent materials cannot be detected when using lasers. Other sensors like sonar sensors can be used as an alternative.

#### *6.2.5. Vision guided localization*

Vision-guided localization is similar to contour localization. Instead of using a LiDAR, a stereo camera is used to make images from which 3D-point-clouds can be built. Each pixel of the camera is converted to a point in the 3D-space which is situated before the camera. This 3D-point-cloud consists of points which represent features in the area seen by the camera. This point-cloud can then be projected onto a 2D-point-cloud which is a 2D projection of these features. An occupancy grid system [91] can be used to represent the local map of the environment. An occupancy grid is a cell decomposed representation of the environment, see Section 7.2.1. The whole area is divided into a grid of small squares. Each square is denoted either as "Unvisited", "Occupied" or as "Unoccupied". Also, the probability of occupation can be used. By project-

ing the 2D-point-cloud of the already seen features onto the occupancy grid, the features seen by the camera are translated into occupied cells. In this way, a map of the environment can be constructed by moving around and continuously projecting the seen 2D-point-cloud onto the occupancy grid. Like in natural localization, the robot also needs an initial map of its environment which can be scanned by a person when exploring the total area using SLAM. A disadvantage of this method is that camera images are sensitive to light conditions which frequently appear in real-life environments.

### *6.3. Sensor fusion*

In practice, the above localization methods are not implemented standalone. Because of noisy sensor data and drift, the uncertainty on the measured position is too large to properly navigate a vehicle when using only one localization type. For this reason, localization methods can be combined with filters or other types of localization methods. The combination of different localization methods and filtering is called sensor fusion [92]. Sensor fusion can be divided into direct and indirect fusion. Direct fusion combines sensor data of different homogeneous or heterogeneous sensors. Indirect fusion combines sensor data with information on prior knowledge of the environment and input.

- In direct fusion, different localization techniques are combined. Two techniques which are not usable on their own because of high uncertainties are odometry and inertial localization. These are frequently combined with other localization techniques to obtain more accurate results. In odometry [87], the robot calculates its new position by knowing its starting position, the distance it already traveled, and the angle it is rotated. By using odometry sensors on the wheels, these distances and angles can be measured. However, this technique cannot be used as a standalone localization technique as it is far from accurate because of the slip of the wheels on different surfaces and other uncertainties. Because of this, odometry is used to combine with other localization techniques to gain more accurate measurements. Inertial localization [83] adds a gyroscope which detects and corrects the smallest change in the heading of the vehicle. Combination of inertial localization and other localization techniques will improve accuracy.

- An example of an indirect technique is a combination of a localization technique with some form of a Kalman Filter [93, 94]. This filter uses a model of the process to make a prediction of a next state using the current state and the properties of the process. This prediction is combined with the sensor measurement of this next state to obtain a more accurate estimation. The filter considers noise on sensor measurement and on the transition from one state to the next. The Kalman filter is a very commonly used technique to improve accuracy in mobile localization.

In practice, sensor fusion is always implemented to have an accurate position estimation of the robot. Without this, the measurements would be too noisy to properly navigate a vehicle.

### *6.4. Conclusion*

During the years, a variety of localization techniques are developed. Flexible systems which are required for the future cannot work with physical circuits which are difficult to adapt. In the review, we only talked about physical and virtual predefined paths. In current industrial systems, navigating on predefined paths is mostly used because of robustness-related issues. Although, it is also possible for an AGV to move freely into a 2D area without being fixed to a predefined circuit. But this is not widely implemented in the industry. To localize freely into the 2D area, also the virtual path navigation methods

can be used. For future AGV systems, virtual path localization techniques are preferred due to the adaptability of the circuit. For this reason, natural and vision-guided localization methods will gain a lot more attention in the future. The disadvantage is that these methods are not very accurate and can be disturbed by a lot of factors like ambient light, vibrations, and uncertainties in measurements. Hence, a combination of these techniques with the very accurate and robust laser localization will be, in the opinion of the author, the future of AGV localization. Another argument why natural- and vision-guided methods are preferred, is because of the possibility to obtain additional information from the environment. Using a camera or LiDAR, an AGV can perceive a lot more of its surroundings. It can for example track objects on the route which can be static or dynamic. Also, object recognition can be added. This to detect hazardous situations, persons, or misplaced objects. [95] proposes a method where camera and LiDAR information of different AGVs are gathered and combined to create and update a global map of the environment. In this way, each AGV can know what is happening in the entire area. Table 3 compares all localization methods.

## 7. Path Planning

A next core AGV task is path planning [96, 97]. We interpret path planning as the static planning task of an AGV whereas motion planning, which is going to be handled in the next section, can be seen as dynamic path planning. Static planning means that a basic collision-free path is computed using known information. No dynamic time-dependent elements are considered, only the known map with obstacles is used. Using this map, the robot knows its free configuration space and the obstacle space. Path planning can thus be defined as the generation of an obstacle-free path, connecting the start point with the goal point, taking into account the geometric characteristics of obstacles and the kinematic constraints of the robot. Path planning consists of two steps:

- Representation of the free configuration space.
- Using a graph search algorithm to search for the shortest path using this representation

Although path planning is used to generate shortest paths, it is also frequently used in AGV-systems to calculate the cost to reach a certain goal. This information is frequently used in task allocation algorithms. In this section, several methods to represent the environment and some search algorithms to compute the shortest path using this representation will be illustrated. An overview can be seen in Figure 13.

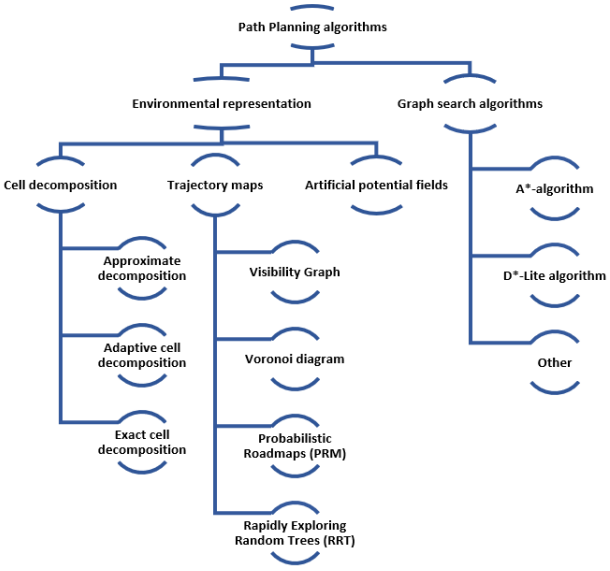



Figure 13: Overview Path Planning algorithms

### *7.1. Desired properties*

The goal of path planning is to generate a shortest path from a start to an endpoint which minimizes an objective function. This objective can be travel time, travel distance, fuel consumption, or combinations of them. Graph search algorithms are used to find a solution which connects the starting point and the goal point by minimizing this objection function. A strong requirement for path

| Localization methods | | |
|---|---|---|
| Method | Advantages | Disadvantages |
| Physical path localization | | |
| Inductive localization | Accurate, simple algorithm | Time expensive to install and modify |
| Optical localization | Cheap, simple algorithm | Circuit not easily adaptable, easily damaged |
| Magnetic tape localization | Cheap, simple algorithm | Circuit not easily adaptable, easily damaged |
| Virtual path localization | | |
| Magnetic spot localization | Easily adaptable circuit, accurate | Time expensive to install and modify, special infrastructure needed |
| Laser localization | Easily adaptable circuit, very accurate | Expensive sensors, special infrastructure needed |
| GPS localization | Easily adaptable circuit | Not quite accurate, special infrastructure needed |
| Natural or contour localization | Easily adaptable circuit, additional information acquisition from environment, no special infrastructure needed | Expensive sensors, not quite accurate, sensitive to reflecting material |
| Vision guided | Easy adaptable circuit, cheap, additional information acquisition from environment no special infrastructure needed | Not quite accurate, sensitive to light |

Table 3: Overview table Localization algorithms

planning algorithms, is that they have to be complete. A search algorithm is said to be complete if it finds a solution or correctly reports that there is no solution, and this in a finite amount of time. Incomplete planners, on the other hand, does not always find a solution when one exists. Another important property is time complexity. Path planning will be calculated a lot of times and also re-planning of paths will occur frequently. For this reason, the time complexity has to be as small as possible.

### *7.2. Representation of the environment*

A path planning algorithm has as purpose to compute the shortest path. To do this, the algorithm first needs a representation of the possible reachable states of the AGV on the environmental map. In the current deployed industrial AGV systems, the paths where these AGVs can move on are predetermined. Thus a circuit which consists of nodes (intersections) and segments (paths between the intersections) is defined and designed. This network of nodes and segments is then used by a search algorithm to search for the shortest path from point A to B. If such a predefined circuit is not available, an algorithm is needed which generates a representation of the configuration space in such a way that possible paths are generated into the full free configuration space. There are several algorithms for this purpose. These algorithms are unfolded next.

#### *7.2.1. Cell decomposition methods*

In cell decomposition methods [98], the total environmental map is decomposed into a grid of cells with a certain size. This can be seen in Figure 14.

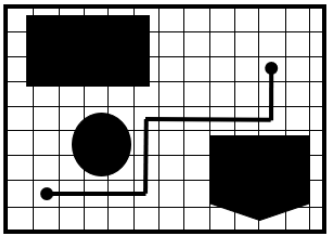


Figure 14: Cell decomposition method

Each cell is either defined as occupied or non-occupied. This is also called an occupancy grid. All the cells which are marked as occupied represent obstacles like walls, tables, or other structures. In the unoccupied cells, the robot can move. The occupancy can also be probabilistic. If a robot defines a cell multiple times as occupied because of what it perceives, the probability that the cell is occupied in reality is larger. Using this technique a representation

of the total area is created, knowing in which cells the robot can move (free configuration space) and in which cells it cannot (obstacle space). This cell-based representation can be translated into a connectivity graph which represents the adjacency relations between cells. A graph search algorithm like an A*-algorithm can then be used to find an optimal path between the cell where the robot is situated and the goal cell. There are some different types of cell decomposition methods:

*Approximate decomposition.* The size of all the cells is predefined and fixed. A disadvantage is that the complexity grows with the dimensions of the grid. And a smaller obstacle than the size of a cell will occupy the whole cell.

*Adaptive cell decomposition.* A large cell size is chosen at the beginning. If a cell is partially occupied by an obstacle, the cell is divided into four equal parts. This is repeated until each cell is either fully occupied or non-occupied. This approach uses less memory but can result in difficulties in dynamic environments where a robot constantly needs to update its map when seeing other obstacles.

*Exact cell decomposition.* The map is decomposed into cells which are based on the map and the locations and shapes of the obstacles. No fixed size of the cells is predefined. The cells take over the shapes of the obstacles so the union of all the free cells represents exactly the free configuration space.

### *7.2.2. Trajectory maps*

In cell decomposition methods, the total area is decomposed into a grid of cells to represent the configuration space. This can be seen in Figure 15. In this approach, an algorithm is used to fill the total area with possible paths. These paths can be generated in different ways:

*Visibility graph.* In this approach [99], a graph of possible paths to move on is constructed by connecting all the vertices of the obstacles present in the area map. These obstacles need to be represented as polygons with straight lines. Using the generated graph of vertex connections, the shortest path can then be calculated by using a simple graph search algorithm. One thing to mention here is that if the robot will move on segments which are made by connecting obstacle vertices, the robot will definitely collide with those objects. This can be prevented by enlarging the obstacles with the size of the robot itself, this is called dilating. In this way, the robot has a graph of all possible paths it can move on without the possibility to collide with any object.

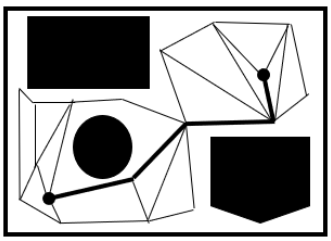


Figure 15: Trajectory map

*Voronoi diagram.* A Voronoi diagram [100] consists of a graph of lines which are equidistant from the two nearest obstacles. This guarantees that the paths to move on are positioned as far as possible from every obstacle. Obstacles here are also represented as polygons. Once the graph is constructed, a start and endpoint are added and connected to the graph and a graph search method is used to find an optimal solution. The AGV can then move on the vertices of the 2D-Voronoi diagram.

*Probabilistic Road Maps (PRM).* A PRM [101] uses random sample points in the configuration space of the robot. These sample points will act as path intersections. If these random samples are in the free space, they are connected to neighboring samples if the line between the two samples is collision-free. This can be the k-nearest neighbors or neighbors within a certain distance. The process of connecting the samples is continued until the graph is dense enough. After the graph is made, start and goal configurations are added to the graph. A graph search algorithm

can now be used to connect the start point and the goal point with the shortest path. When the number of sampled points reaches infinity, a non-optimal path will certainly be found when there exists one.

*Rapidly exploring random trees (RRT).* In the RRT [102] algorithm, a tree is randomly expanded from the start node. An edge is only added to the tree if it does not cause a collision. Using this technique, an area can be rapidly covered by a tree of paths where an AGV can move on. An algorithm where two trees grow towards each other can also be used. One starting from the start and one starting from the goal, this is called bi-directional search. This is used to fasten up the process. These two trees will then be linked together. If the start and goal node is connected to the tree, a graph search algorithm is used to find the shortest path. If the number of expanded nodes approaches infinity, a non-optimal path will certainly be found when one exists.

#### *7.2.3. Artificial potential fields*

Using artificial potential fields [103], a robot finds a path where it can move on by a superposition of all potential fields the robot senses. There are repulsive fields emitted by obstacles with a force inversely proportional with the distance to the obstacle. There are also attractive forces which are emitted by goal states. By using this method, the robot may get stuck into local minima.

### *7.3. Graph search algorithms*

In the previous part, alternative representations of the environment are given. These representations are actually graphs with nodes and edges, which are further used by a search algorithm to compute the shortest path. Here the authors want to notice again that these generated representations are only used in situations where predefined paths are missing. But in all current industrial AGV systems, these AGVs move on predefined circuits. These circuits are used as a base to do path planning. Some of the graph searching algorithms are unfolded next.

#### *7.3.1. A*-algorithm*

The A*-heuristic algorithm [104] is one of the most popular classical graph search algorithms in calculating the least-cost path on a weighted graph. This algorithm uses a weighted graph with nodes as locations and edges between these nodes containing the cost to go from one node to another. Also, a list of unvisited nodes and a list of visited nodes are maintained. The algorithm makes use of a heuristic to find an optimal solution much faster, this can be an estimated cost from a node to the goal node. The algorithm starts from the initial start node and works towards the goal by visiting and evaluating each neighboring node in the unvisited list. It is an efficient and complete algorithm but has high memory usage. A* guarantees to find the optimal shortest path if there is one.

#### *7.3.2. D* Lite-algorithm*

D* Lite is an extension of A*. In contrast with A*, D* Lite [104] works in the opposite direction which is from the goal to the start. Also different is the use of an extra parameter $rhs$ which is introduced to give info about the cost of one-step ahead. Also, a heuristic is used here but one which estimates a cost to the start. This algorithm is an incremental heuristic search algorithm which re-uses trees from previous searches. This to speed up the search process. The D* Lite-algorithm has good results in large and complex areas. It plans shorter paths much faster than A*-algorithms. It is less effective than A* in simple and small areas. Whereas A* cannot do re-planning, D* Lite can re-plan because it keeps previous information. Because of this, D* Lite algorithm is the most widely used path planning algorithm.

#### *7.3.3. Other algorithms*

The previously mentioned algorithms are the most common in path planning for AGVs. But there are more gen-

eral optimization algorithms which can be used. Some which are already mentioned in the section about task allocation like Tabu Search, Genetic Algorithm, Particle Swarm Optimization, Ant colony algorithms, and Simulated Annealing can be used. But the graph search algorithms like A* and D* Lite, are the most common for mobile robotics.

### *7.4. Conclusion*

There are several methods to represent environments and several search algorithms to find a shortest path given the graph. Cell-based and generated trajectory map approaches are more used for environments when there are no predefined paths and where a robot can move freely into the free configuration space. In currently deployed AGV systems which work centralized, the representation of the environment is predetermined and graph-based (a collection of nodes and edges). Using this graph-based circuits, a graph search algorithm is used to find the shortest path from start to goal. Future decentralization requires more flexible methods while ensuring predictable behavior. If AGVs can move freely in the factory without following any predefined circuits, then the behavior of the AGV will not be predictable for employees anymore. So in the opinion of the authors, these predefined graphical-designed graph-based circuits will still be used as a base in future systems together with the more dynamic path planning types. This so that the AGV can mainly move on predefined fixed paths but can leave the path and move freely when, for example, something is blocking the path. The same graph searching algorithms to find the shortest paths and to find costs to reach a given goal will be used as they have proven their utility in current systems.

## 8. Motion Planning

Section 7 describes static path planning where a basic path is constructed only considering static obstacles like walls and racks. But in reality, this static path is not collision-free. During the execution of the path, the AGV can face obstacles where the static path planner does not know about. These obstacles can be unforeseen static obstacles, people, and other moving AGVs. Obviously, also collision with these features has to be prevented. Besides collisions, also deadlocks [105] need to be prevented. A deadlock is a situation where an AGV has no possible actions anymore. It can move forwards neither backward. The modification of a static predetermined path to avoid collisions and deadlocks is called motion planning [96].

Centralized motion planning is currently used in most of the industrial AGV systems. Collision- and deadlock-free trajectories are planned by the central controller for all the robots simultaneously. Central motion planning makes use of global information. The central unit knows all the AGV positions, their goals, and the static paths they are about to execute. An optimization process uses all this information to search for a solution space for an optimal solution. Although this central optimization approach is very powerful as it can consider this global information, it is constrained by the computational time requested for a real-time motion of the robots, which increases with the number of used AGVs. Also, the central unit acts as a single point of failure which restricts robustness of the system. In distributed motion planning, an AGV is self-responsible for avoiding collisions and deadlocks. After a fixed predefined path is calculated minimizing a certain cost, the AGVs use their local information and perception to react upon unforeseen circumstances while executing this path. This is exactly how humans control their movement: we can only consider what we see around us and act upon that to avoid collisions or deadlocks. The AGV can, on top of that, also coordinate with neighboring devices to obtain more local information useful for motion planning. This way of motion coordination is much more robust and scalable than centralized planners as an AGV locally computes a plan using local information. In this way, the informa-

tion and intelligence is distributed which eliminates the single point of failure and the limit of AGVs which can be added to the system.

**Remark:** One of the properties of distributed control is that a device only uses local information. However, in motion planning, the AGVs also can get access to global information generated by all AGVs independently. [95] shows a method in which the perceptions of each AGV are collected and combined into one shared cloud map of the environment. AGVs can individually update this map and retrieve useful information from it. In this way, it is still possible to have access to global information which can make distributed motion planning much more efficient and robust. In the further sections, the authors divided this motion planning into three parts:

- Collision Avoidance
- Deadlock Avoidance
- Zone Control

### *8.1. Collision avoidance*

In collision avoidance, collisions with static or dynamic objects which cross the trajectory of an AGV are avoided. The most simple way to do this is to use a safety scanner which keeps up a safety zone. When objects come into this safety zone, the AGV slows down and finally stops. After the path is cleared, the AGV continues its path. This is widely used in the industry. More advanced methods can be used which try to move around an object or plan an alternative route. In the next sections, we cover central collision avoidance algorithms as well as decentral algorithms.

#### *8.1.1. Centralized collision avoidance*

To avoid collisions in the system, the central computer uses global information to do an optimization. Using all the available information like positions, goals, and presumed paths for each AGV, an optimizer can search the solution space looking for an optimal solution which is a collection of collision-free trajectories generated for each AGV. When a particular solution is generated, time intervals in which the AGV will occupy a segment can be determined, and this for all AGVs. For each of these segments, there can be determined if two AGVs occupy the same segment in the same time interval. If this is the case, the found combination of generated trajectories is discarded and a new combination is generated. Instead of generating a totally new solution, a new feasible solution without segment occupancy by two AGVs at the same time can be found by modifying the trajectory of one of these AGVs. Another totally different trajectory can be found [43, 106], or one of the vehicles can be slowed down[107]. This process continues until the central controller finds a set of collision-free trajectories for each AGV in the system. To execute this searching process, the central unit needs the positions of all AGVs. But as these change continuously, this causes the need for this process to repeat frequently and consuming a lot of time.

In current industrial systems, this central collision avoidance process is mostly included in the task allocation optimizations [43, 107, 108]. For a certain tasks-robots allocation, the motion planner can output a feasible solution for all the paths which needs to be done to perform the allocated tasks. This generated set of collision-free, trajectories can be used to calculate the total traveled time/distance of all the AGVs which can be used as a fitness measure for the task allocation optimization. By including this collision avoidance into the task allocation, a heavy and complex optimization is obtained. This asks for a lot of computing power and computation time. If something in the situation changes, for example, if an AGV malfunctions, then this hard optimization needs to run again.

It is obvious that this method is not scalable. If a lot of

AGVs are in the system, this becomes computationally too expensive. A second drawback is the fact that these computations are very time consuming causing a delay with respect to the real situation. This can cause that the optimization does not respond to very dynamic occurrences which limits flexibility. A third drawback of centralized collision avoidance is that the central planner can only consider collisions with other AGVs and with objects it knows from the static environmental map it has. But it cannot consider collisions with other dynamic objects like humans or objects which are not on this map. For a flexible and robust behavior, there is a need for more information acquisition. If the central controller also needs to consider all dynamic features, it will be probably overloaded. This is a reason for the authors to prefer a more decentralized collision avoidance in future systems so an AGV can react upon these dynamic features locally using only local information which is not that computational hard.

In the next section, decentralized collision avoidance is presented. These algorithms act upon what they perceive locally, and can thus better adapt to changing circumstances.

*8.1.2. Decentralized collision avoidance*

In decentralized collision avoidance, the AGV reacts upon what it perceives locally to avoid collisions. If it detects a possible collision on its path, it can decide to slow down and to stop until the path is cleared. Or it can try to move around the obstacle locally. Due to this local reactive behavior, these algorithms are more suitable than the computational-heavy and static central collision avoidance controllers for future AGV systems.

*Forward sensing.* This is the most simple decentralized collision avoidance method in which AGVs keep up a safety zone using an on-board safety sensor which monitors every object in front of the vehicle. When an object or person comes into this zone, the AGV will first decrease speed and can eventually come to a stop. When the object or person is removed, the AGV resumes its movement. The advantage of this method is that no environmental map is needed for this method, only the onboard sensors can be used to react upon dynamic obstacles. The algorithm will not really cause the robot to avoid obstacles but rather stop safely to not collide with anything. This should be implemented on every AGV as a basic safety layer which prevents colliding with obstacles in any case, also when other collision avoidance algorithms would fail.

*Re-planning using A* or D*.* When the robot is executing its trajectory and an unforeseen object crosses the path, the calculated path can be re-planned [104]. The same algorithms for initial path planning can be used. If A* is used, a totally new path has to be calculated as A* does not keep historical data. This new path is calculated using the map, the goal position, and the new start position. As computing this new path from scratch takes a long time, using A* is not very flexible. A better alternative for re-planning is the D* Lite algorithm. This algorithm does keep historical data in its memory. The re-planning can thus be done much faster and efficient which makes this algorithm much more flexible and convenient for the job. For this re-planning, a representation of the environment map is used. This map also needs to contain dynamic obstacles. Otherwise, the robot will only re-plan considering static obstacles known by the map as this map is the only information source of this re-planning method. Cloud-based maps in which all AGVs update the map from their own perceptions in parallel [95], can be used.

*Local deviation from the path.* This technique [109] uses local data to calculate a local deviation from the circuit. It consists of four steps: a safety check, leaving the road map, overtaking the obstacle, and return to the road map. During the safety check, the AGV monitors if an object is present on the robot's path. If there is an obstacle, the AGV builds a new segment which leaves the roadmap using

a lane-change maneuver curve. This curve is a polynomial path which starts from the current position of the robot and ends on a line parallel with the current path. It is obtained considering maximum curvature, steering-rate, a minimal deviation from the map and a safe distance to the obstacle. Thus the kinematics and dynamics of the robot are considered. To overtake the object, a line parallel to the roadmap is maintained without decreasing the distance to the road map. To return, another lane-changing curve is calculated using the same technique as in leaving the road map. This method does not need an environmental map with obstacles, it can be implemented using only the circuit on which it moves and the onboard sensors. This is a very effective method in safely overtaking a static object but is more difficult in overtaking dynamic objects.

*Virtual force fields (VFF).* The Virtual Force Fields [110] method uses an occupancy grid representation as mentioned in Section 7.2.1. The occupied cells will repel the robot away with repellent forces which are dependent on the concentration of occupied cells. The force is inversely proportional to the square of the distance between the robot and the cell. The robot will move along the gradient of the total field constructed by the superposition of all the repellent force fields. Disadvantages of this method are the difficulty with obstacles which are too close to each other, because of the repellent forces on both sides, the robot will not move in between. Even though there is a place for the robot to move in between. This method does not deal with kinematic and dynamic constraints. An environmental map with obstacles is used. The same properties are present as in re-planning using A* or D*-Lite algorithms. The map should also contain dynamic features to have flexible collision avoidance.

*Vector field histogram (VFH).* The VFH-method [111] solves the problem in the previous section where the robot does not move between obstacles which are to close to each other. This approach also uses a 2D-histogram grid to represent the environment. But here, the cartesian histogram grid is reduced to a 2D polar histogram which is built around the position of the robot at the time. This histogram shows the obstacle density seen from the robot perspective. In each possible heading direction, an obstacle density is calculated. The direction of the robot is then chosen based on the least concentration of obstacles. This algorithm also does not deal with kinematic and dynamic constraints. For this reason, an alternative algorithm (VFH+) [112] is proposed which employs a threshold hysteresis to improve the shape of the trajectory. A cost function is used to choose the best direction in a space of possible directions provided by the polar histogram. This method also considers the vehicle width by enlarging the cells containing obstacles. This method can work with an environmental map as well as with on-board sensors to detect the obstacles in front of the robot.

*Dynamic windows approach.* The algorithm [113] makes use of a solution space which consists of all feasible velocity vectors which can be commanded to the robot. The velocity vectors are actually arcs defined by a velocity vector $(v, w)$, where $v$ is a linear velocity and $w$ an angular velocity. A time interval of $\tau$ is defined which can be used to calculate the forward and rotational displacement in this time interval when commanding a particular velocity vector to the robot. The solution space of feasible velocity vectors is reduced with all the velocity arcs which are not physically reachable by the AGV. This for example due to kinematic or dynamic constraints ($v_{min}$, $v_{max}$, $a_{min}$, $a_{max}$). Also, velocity arc vectors which cause collisions with surrounding objects in the time interval are removed. What remains is a solution space containing all velocity arc vectors which are reachable by the AGV and which do not tend to a collision. Once this solution space is defined, an optimization algorithm can search for the best velocity vector optimizing a certain objective. This vector can then be commanded to the robot which results in a certain dis-

placement. When iterating this process every $\tau$ seconds, a collision-free trajectory is followed by the robot. This algorithm also deals with dynamic and kinematic constraints. This method can work with an environmental map as well as with on-board sensors to detect the obstacles in front of the robot.

*Optimal Reciprocal Collision Avoidance (ORCA).* ORCA [114] is a collision avoidance algorithm where each robot knows its own position and velocity, and the position and velocity range of other robots. Each robot calculates a velocity space of velocity vectors $(v, w)$ which will definitely cause collision within a certain time $\tau$ (similar to Dynamic Window Approach). A Cartesian space is constructed which represents the total area where the robot can get in a collision within time interval $\tau$ for each velocity which would be commanded to the robot. The geometry of this space is constructed using the geometry details of the robots (for example radius), the position of the other robots, and the $\tau$-constant. Using all the velocities which are not in this space, it will be certain that the robot does not collide for at least the period $\tau$. The robot creates a boundary between allowable and non-allowable velocity vectors to prevent choosing a vector which will cause a collision. From the remaining velocity space, the velocity is chosen which is the closest to a preferred velocity stated by the motion planner and which does not exceeds maximum velocities. By linearizing the constraints, this problem of searching for an optimal speed can be done using linear programming. Every time step, the robot senses the positions and velocities of other robots, compute its boundary space, selects a new velocity outside this boundary using linear programming, and applies the velocity to its actuators. This is actually the opposite of the Dynamic Window approach where feasible non-collision velocity vectors are calculated instead of velocity vectors which cause a collision. This method is designed more specifically to avoid collisions with other robots in a free-moving area. However, it can also be used to avoid other objects. This algorithm also deals with dynamic and kinematic constraints. This method can work with an environmental map as well as with on-board sensors to detect the obstacles in front of the robot.

*Predictive models.* This approach mimics human behavior. Imagine being in a crowded metro station with hundreds of people having their own goal in mind and searching for the most optimal path through this bunch of people. Although this is a very complex situation, almost no collisions happen. But what actually are the steps we take to fulfill this task? Every time step we perceive everyone around us and we try to predict where they are going. This is possible as the probability that someone makes an abrupt movement is very rare. Dependent on all these predicted velocity directions of all those people, we are going to adapt our path and thus adapt our own velocity direction or vector. An example of a predictive model is a model for collision avoidance which uses deep reinforcement learning [115]. By perceiving the locations of the surrounding moving elements time step by time step, a reinforcement learning algorithm is going to predict the appropriate action, which is a velocity vector. This action can be chosen from an action space, which consists of velocity vectors from speed zero to the maximum speed and this in all directions. The robot has a value function which gives as output this velocity vector given the sequence of perceptions as input, and this step by step. This value function is learned by performing random actions and looking at the results of those actions. The robot is rewarded when reaching its goal but punished when coming too close to an obstacle. The coefficients of this value function are learned by a deep learning network which is trained using data retrieved from the previously described ORCA method. A large number of trajectories generated by this ORCA algorithm are used to train the network using deep learning.

*8.1.3. Conclusion*

We saw that centralized collision avoidance is very computationally heavy and is restricted to static objects. Considering also dynamic objects would make the optimization process even more computationally heavy introducing delays which result in an asynchronous response in comparison with the real-world. Combining this central motion planning in task allocation optimizations makes it even more heavy and complex. This causes central motion planners to be highly ineffective when operating large fleets with lots of constraints. Decentralized motion planning, however, is much less computationally heavy. It mainly considers its local environment and can have a reactive behavior upon these local perceptions. Collision-free paths can be planned using on-board sensors, an environmental map, or both. Most of the mentioned algorithms are designed for robots which move freely into the area. This is a feature which can cause discussion due to safety-related issues. It can be very dynamic for an AGV to move freely but due to safety-related issues, this can be an unwanted behavior. When operating together with employees, it can be preferable for the employees to be able to predict where the AGV will move. When moving on fixed paths, this is not a problem. But when moving freely, the future position of the robot is unpredictable. For safety reasons, industrial AGV users prefer mostly that the AGV moves on predefined circuits. This said, re-planning using D*-Lite (using the circuit as a graph) and local deviations from the path are very suitable algorithms to perform predictable collision avoidance. In re-planning, an AGV will always stay on track, and in local deviation, the AGV will locally leave the path to return to it behind the obstacle. Using these algorithms gives a very good trade-off between flexibility and safety/predictability in industrial situations. And of course, in both central as decentral systems, the forward sensing should always be implemented as a last safety measure. In Table 4, a comparison between all collision avoidance methods is shown.

*8.2. Deadlock Avoidance*

In literature, there are several ways to prevent deadlocks. First, the layout of the circuit where an AGV can move on can be designed to reduce deadlocks. This option is more related to the design of an AGV system. As the paper only focuses on the control of AGVs, we will not cover this part as we are not focused on design. Second, the environment can be divided into several control zones in which the maximum amount of AGVs can be kept. This is also used to avoid collisions and is covered in the next section 8.3. And thirdly, some routing strategies can be developed to prevent deadlocks. These are algorithms which incorporate traffic rules or consensus between vehicles to avoid deadlocks. This is covered in the following sections. We make a distinction between central and decentral deadlock avoidance. Algorithms for both approaches are described next.

*8.2.1. Central deadlock avoidance*

Central deadlock avoidance is considered into the large optimization process together with task allocation and collision avoidance as mentioned in Section 8.1.1. The way this central deadlock avoidance can be included in the optimization can be done in different ways. Petri nets [116, 117] are a frequently used approach for deadlock avoidance in central motion planning. A Petri net, or a place/transition net, is a graphical tool to represent systems with concurrency. It is a directed bipartite graph of nodes, transitions, and arcs. A node can be a condition or a state of the system, and a transition can be an event that may occur. The directed arcs describe the relationship between nodes and transitions. It clarifies which nodes are pre- and/or postconditions for which transitions. These nets can be used to model the AGV system layout and the paths that the vehicles follow [118]. Li *et al.* [119] give a review on the deadlock avoidance in automated manufacturing using Petri nets. Other algorithms use a strategy to detect a cyclic-waiting situation [120], using a graph the-

| Collision avoidance methods | | |
|---|---|---|
| Method | Advantages | Disadvantages |
| Centralized collision avoidance | | |
| General | Optimal for small fleets | Lack robustness, performance, scalability, and flexibility for large fleets |
| Decentralized collision avoidance | | |
| Forward sensing | Simple, robust, good for final safety check | None |
| Re-planning using A* or D* | Simple, fast | Needs environmental map |
| Local deviation from the path | Safe overtaking procedure | Not for complex avoidance (overtaking dynamic objects) |
| Vector Field Histogram (VFH) | Simple, effective | More for free-navigation, difficulties with small distant objects |
| Dynamic Windows approach | Simple, effective | More for free-navigation |
| ORCA | Simple, effective | More for free-navigation |
| Predictive models | Good performance in dense situations | Too complex for simple/predictable collision avoidance, takes a lot of time to train |

Table 4: Overview table Collision Avoidance algorithms

ory [121], or using a matrix-based deadlock detection algorithm [122]. All these central optimizing deadlock avoidance controllers are very time consuming and complex algorithms which lack robustness and flexibility. Because of their working principle, they can only detect a deadlock a few moments before it would occur, after which they attempt to resolve it. In the future flexible systems, this is not permissible. Deadlocks should be avoided instead of detected and resolved. Because of these aforementioned drawbacks, centralized deadlock avoidance is not suitable for future AGV systems.

*8.2.2. Decentral deadlock avoidance*

In decentralized deadlock avoidance, AGVs try to avoid deadlocks using their local information. So only what they can observe, can be used to avoid deadlocks. This in contrast with centralized optimizers where the central unit has access to all the positions and future planned segments for each AGV. Thus also in this part of the AGV control, decentral approaches are more suitable than the previously reviewed central algorithms because of their decentralized and local-perceiving nature. Markus *et al.* [123] proposes such a distributed algorithm which consist of a collision avoidance algorithm, a deadlock detection algorithm, and a deadlock resolution algorithm. When the distance between two robots drops below a certain value, the robots initiate a coordination link. One of the two robots is the coordinator and one is the partner. This link is used to detect and resolve deadlocks and to avoid collisions. The link either permits the robot to move on or not. A similar approach [124] makes use of a two-layer architecture for path planning where the coordination among AGVs is based on a negotiation of shared resources. For each section on the static path, the AGV checks whether the section is available or occupied by another AGV. If more AGVs want the same segment, then an auction process is initiated. The winner can move on the segment. In [125] presumed collision-free trajectories are planned for each AGV by a central computer like in section 8.1.1. These presumed paths are exchanged with neighboring AGVs so they know each AGV's intentions. Each AGV then uses the presumed trajectories of neighbors to locally compute a collision-free trajectory according to a priority policy. Dario *et al.* [126] proposes another decentralized deadlock avoidance algorithm based on a shared resources protocol and a re-planning strategy. The coordination is based on a set of rules like in urban situations. The rules are used to get access to local resources. Although these decentralized algorithms incorporate good characteristics, they are too

rule-based. These rules need to be hard-coded into the algorithms which prevent the algorithm to behave well in all situations. In the opinion of the authors, there has to be put more attention towards decentralized non-rule-based motion planning in future AGV research. Deadlocks are one of the main bottlenecks in AGV systems which cause a lot of failures in the system. If a deadlock occurs, most of the time the whole system gets blocked which causes high time loss. For this reason, a good decentral deadlock avoidance method which manages traffic efficiently independent on the scale of the system is needed for future AGV systems.

### *8.3. Zone Control*

Zone control is the management of a particular zone to prevent a limited amount of AGVs to be exceeded into this zone. It is one of the most effective strategies to prevent collisions with other AGVs. The use of zone-control eases the avoidance of collisions and deadlocks. This technique divides the circuit where the AGVs can move on into non-overlapping zones in which the maximum amount of vehicles is limited. Vehicles which want to enter a full zone have to wait or replan their route. Traditional techniques are fixed zone strategies where the zone areas are fixed. More new techniques make use of dynamic zone control techniques in which the size and arrangement of zones can vary. Li *et al.* [127] presents a traffic control scheme based on a novel discrete-event zone-control model. In [128], some zone design methods are covered and a dynamic zone strategy is proposed. In [129], another dynamic zone control is proposed. The objective here is to maintain the same workload for each vehicle. Zones are redesigned during the operation to avoid differences in workload between the zones. Fanti *et al.* [130] proposes another zone-based control where the guide paths are subdivided into disjoint zones representing intersections, straight lines or workstations. Only one vehicle can occupy a zone at a time and the permission to enter a zone must be given by the control. This zone control can definitely be a help to ease collision avoidance and deadlock avoidance in general. They can thus certainly be used in a distributed control of AGVs in future systems.

### *8.4. Conclusion*

Besides task allocation, motion planning is another very complex core AGV task. It covers collision avoidance, deadlock avoidance, and zone control. Centralized approaches use global information for an optimization process to do motion planning. Because they have access to all the information, they can prevent collisions and deadlocks by considering the whole movement of the fleet. They take a snapshot of the current AGV configuration and continuously optimize the allocation concerning the current situation. This approach is widely used in today's industrial AGV systems. However, this approach incorporates no flexibility and is very time-consuming. More flexible is when an AGV can react from the moment it encounters a problem. This is exactly how humans prevent collisions and deadlocks. By perceiving the local environment, we can act properly to changes at the moment. This is what decentralized motion planners do. They perceive their local surroundings and act upon them. The authors believe that because of these properties, decentralized motion planners will be used more frequently in future industrial systems as these will become larger and more complex. Besides, the authors like to mention that in future AGV research, more attention should go to decentral deadlock avoidance to manage traffic more efficient with a performance independent of the size of the system. Also interesting is that current algorithms are based on a device-to-device principle. For future systems to be robust and flexible, more research should focus on device-to-infrastructure and infrastructure-to-infrastructure communication to get access to more than local information. By interacting with the environment, like humans "communicate" with traffic lights, for example, traffic in AGV

control can be managed more robust in contrast to only AGV-to-AGV communication. In the opinion of the authors, this is definitely a new point of view in AGV motion planning to consider in future research towards robust and large-scaled AGV control.

## 9. Vehicle Management

Vehicle management is the simplest core AGV task in that sense that it does not require complex algorithms or techniques. It monitors battery status, error status, and maintenance status. This status will cause constraints on the possibility to perform tasks and thus need to be considered at the task allocation level. Using a centralized control, the central controller takes all the vehicle management statuses into account. If it knows for example that the battery life of a vehicle is not enough to execute a certain task, it will give the vehicle another less energy-asking task or it will send the AGV to the charging station. When the central unit receives error statuses form an AGV, it will not involve the AGV into the task allocation optimization anymore. Instead, it will send the AGV to maintenance. Using a distributed control, the AGV monitors its own parameters. If it sees that its battery capacity is low, it will, in a market-based approach for instance, not bid on an energy-expensive task but will directly head to a charging station. In general, the status of an AGV will affect the bid calculation in decentralized auction processes for task allocation. When an AGV is in error status, it can decide to reassign its local tasks to neighboring AGVs so that no tasks need to wait. The AVG acts independently upon its own statuses in an intelligent way. Thus again for flexibility reasons and for distribution of computation, the authors put forward a distributed vehicle management to be used in future AGV systems.

### *9.1. Resource Management*

Error statuses or maintenance signals cannot be controlled in that sense that they occur unexpectedly. For the control algorithms, it is just a matter to be able to address them well if they occur. For this reason, we will not pay more attention to these features as this is a review of AGV control. Battery management, on the other hand, can definitely be controlled. It can have an impact on the total performance and is thus, according to the authors, an important and relevant vehicle management task in AGV control. However, it seems that in literature, there is little attention going to this battery management. In her survey, Iris F.A Vis [3] states that battery management is hardly addressed in AGV research. She states that including this resource management into the decision making of an AGV and into the routing of AGVs is a relevant thing to do in future research because it can certainly have an impact on the overall performance. Also, another review on the design and control of AGV systems [2] endorses the same statements that battery management is usually omitted in research. However, McHaney [131] proved already earlier that the performance of an AGV system is clearly influenced by resource management. He distinguishes three different types of charging schemes which are:

- Opportunity charging in which an AGV uses its idle time to charge,
- Automatic charging in which an AGV runs until its battery level drops below a certain limit,
- And combination charging which is a combination of the previous two

Also, the way batteries can be recharged can differ. The most common schemes for battery charging of AGVs are battery swapping and automatic charging [132]. In battery swapping, the AGV swaps its empty battery for a fully charged one. In automatic charging, the AGV charges itself at a station while the battery stays inside the AGV. Ebben (2001) [133] also saw the need for efficiently managing resources. In his paper, he suggested four heuristics for routing the AGVs towards a battery station if its resources drop below a certain threshold:

- Select the closest charging station
- Select the first station on the current route
- Select the furthest reachable battery station on the current route
- Select the battery station that will cause minimum delay considering both travel time and waiting time in a queue

Qazi Shaheen Kabir et al. (2018) [132] made a comparison between the different routing heuristics for battery management from Ebben. They conclude that the heuristic of choosing the battery station which minimizes the total travel time and waiting time when the battery level drops between a certain threshold perform the best. In another article, Quazi Shabeen Kabir et al. (2018) [134] propose a method to increase manufacturing capacities through battery management. They investigate how the duration of battery charging can be varied to increase the flexibility of a manufacturing system. They conclude that a more frequent charging of the battery increases the productivity of a manufacturing system significantly.
If it comes to resource, and more specific, battery management in AGV systems, most of the articles are written on how to manage the battery life cycle. Takehito Kawakami et al. (2011) [135] wrote an article about the battery life cycle management in which they analyze the effect of the AGV operation modes (load, unload, move, etc.) on the battery life and its deterioration. They propose a methodology for planning a battery strategy that minimizes battery-related costs.

Because of the clear demand on research towards resource management in AGV systems, the authors of this paper proposed a decentral resource management approach [136] in which an AGV handles a more efficient charging approach. In future research, we will build further on this.

*9.2. Conclusion*

The authors see resource management as the most relevant part of vehicle management. Resource management is hardly addressed in literature but is very relevant to study as it can have a significant impact on the performance of an AGV system. Literature reveals that resource management decisions do have an impact on the system performance, but is not conclusive on how to approach. Continuing their decentralized preferences in AGV control, the authors also see the need for a distributed vehicle management in future AGV control. As intelligent systems, they need to care for their own resources and responds properly to their own statuses. The distributed computation of decisions upon there resources will benefit the flexibility and robustness of the system.

## 10. Discussion

In our paper, we clearly want to take some steps towards Industry 4.0 if it comes to AGV control. The current employed AGV fleets operate in a classical central and hierarchical architecture where all AGVs act as slave devices which are authorized by a fleet manager, which is further mastered by an order managing system. We strive for a more heterarchical architecture, where all AGVs have an equal role and have the intelligence to operate as master of their own actions. Recently published standards on Industry 4.0 components [4] attempt to present a reference architecture model for Industry 4.0 architectures. This model consists of a three-axis map which shows how to approach the issue of Industry 4.0 in a structured manner. Our vision on how AGV fleets should operate in the future maps perfectly onto this vision of this new standard. Due to the decentralization of intelligence, the hierarchy will fade which will result in participants interacting across all the hierarchy levels. AGVs will not be seen as "dumb" devices anymore but will incorporate their own control while communicating among each other and with all other de-

vices in the manufacturing system.

This said, the authors believe that decentralization of control is the way to go to overcome the current present limitations of centralized control and to provide control architectures the ability to grow and improve. The authors hereby do not claim that decentralization of control directly implies a fully decentralized or anarchic control architecture. The authors believe that there are a lot of gradations in between full central and fully decentral control of which all of them should be considered and evaluated. Users or manufacturers of AGVs should decide for themselves to which level of decentralization they want to migrate. Depending on the scale of the fleet, the demands on the system, and the type of tasks the AGVs need to fulfill, custom hybrid systems will arise which find a good balance between central and decentral control. This results in an important research question for the future how to gradually transform an operating central AGV system into a more distributed system. Industrial AGV systems are largely centralized while pushing the boundaries of this architecture. As these centralized controllers are deeply rooted into the industry, companies are not willing to radically change their total AGV control. Complete decentralized architectures exist with proven performance but are not attractive for companies because of the huge barrier to radically change the complete architecture. Also, the authors do not believe that the transition from dumb devices to fully independent AGVs can be done in one single step. Thus there is a need for research which comes with methods to distribute control architectures gradually. There is a need for a gradual transition which can migrate from a central towards a more hybrid, and eventually to a distributed architecture. For this reason, the authors divided the total AGV control into five discrete core tasks. This separation should make it easier to practically distribute each control task of the AGV gradually. This has as effect that there can be decided which task should be distributed and which task benefits to be more centralized.

The physics of the next generation AGVs does not need large changes. All sensors and actuators which are now present on standard AGVs are suitable to meet the future Industry 4.0 requirements. One of the biggest challenges related to the gradual change in control architecture will be to communicate each device its intentions to other devices to improve the global optimality and to avoid only self-optimizing agents. Robustness and flexibility will result directly from the decentralized architecture as also the ability to increase the complexity of the optimizations (e.g. by considering uncertain environmental models [137, 138]) will do. But as we already spoke of a trade-off between flexibility and optimality, the biggest challenge will be to make the performance as good as possible by interconnecting the whole system.

## 11. Conclusions

This paper covers the current and future-potential state-of-the-art techniques and algorithms to control an AGV system for a central and decentral architecture. We decomposed the total AGV control into a set of five core tasks and reviewed for each of those core tasks the available algorithms and techniques available in literature. The review revealed that a lot of work is done to four out of the five core tasks. A lot of central as well as decentral algorithms and techniques are available in literature to control an AGV fleet according to both approaches. Only the work on vehicle management, though very relevant, is limited. In the paper, we did not only review each algorithm, but we also clarified which algorithms are suitable for future decentralized AGV systems:

- Looking at task-allocation, we can ascertain that optimization-based solutions which attempt to use global information are not suitable to completely manage the allocation of tasks because of the high

computational effort for large and complex AGV systems which harms the allocation performance. Market-based approaches, on the other hand, are suitable due to their distributed properties which directly result in flexibility, scalability, and robustness. These market-based approaches alone will not have sufficient performance: besides, consensus between AGVs and an extra local optimization can definitely improve performance making this combination a very suitable task allocation method for future large and complex AGV systems.

- For localization, more flexible methods which do not use physical predefined circuits are preferable. The virtual path localization methods will be used more frequently in decentralized systems. The authors see a combination of the very accurate laser localization and the very flexible and information-rich contour/vision-based localization as the way to go for localization in the future.
- Regarding path planning, predefined graphical-designed graph-based circuits will still be used together with the more dynamic path planning types. This to ensure that the AGV moves on predefined fixed paths for predictability, but that it can leave the path and move freely to avoid something which is blocking the path.
- For motion planning, the authors saw that current algorithms are mostly based only on AGV-to-AGV interaction. In our opinion, motion planning, and especially deadlock avoidance, could be made more robust for large-scaled systems when more AGV-to-infrastructure and infrastructure-to-infrastructure interaction would be involved in this control. This new point of view in motion planning is definitely worthy to study in the future.

In the practice, we see that current AGV systems almost all work fully centralized but also discern a clear interest in decentralization as an element to meet future requirements like flexibility, openness, scalability, and robustness. The industry 4.0 paradigm will push AGV manufacturers to think about decentralization as an essential element to deal with these requirements as systems will become ever larger and more complex. The authors believe that more decentralized and market-based techniques are suited to introduce flexibility, robustness, and scalability in the total system. The abundance of literature teaches us that already a lot of algorithms exist to that effect in lab environments but that a practical implementation of distributed control still has a way to go.

## Acknowledgments

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

## References

[1] G. V. Research, Automated Guided Vehicles Market (2016) 255–270.

[2] T. Le-Anh, M. De Koster, A review of design and control of automated guided vehicle systems, European Journal of Operational Research 171 (1) (2006) 1–23.

[3] I. F.A. Vis, Survey of research in the design and control of automated guided vehicle systems, European Journal of Operational Research 170 (3) (2006) 677–709.

[4] DIN SPEC 91345, DIN SPEC 91345:2015-07: Reference Architecture Model Industrie 4.0 (RAMI4.0) (2015).

[5] L. Monostori, P. Valckenaers, A. Dolgui, H. Panetto, M. Brdys, B. C. Csáji, Cooperative control in production and logistics, IFAC Proceedings Volumes (IFAC-PapersOnline) 19 (April) (2014) 4246–4265.

[6] 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.

[7] C. Blesing, D. Luensch, J. S. B, B. Korth, Concept of a Multi-agent Based Decentralized Production System for the Automotive Industry, in: Lecture Notes in Computer Science, 2018, pp. 19–30.

[8] C. A. Parker, H. Zhang, Cooperative decision-making in decentralized multiple-robot systems: The best-of-N problem,

IEEE/ASME Transactions on Mechatronics 14 (2) (2009) 240–251.

[9] C. Schwarz, J. Sauer, Towards decentralised AGV control with negotiations, Frontiers in Artificial Intelligence and Applications 241 (2012) 270–281.

[10] R. Walenta, T. Schellekens, A. Ferrein, S. Schiffer, A decentralised system approach for controlling AGVs with ROS, 2017 IEEE AFRICON: Science, Technology and Innovation for Africa, AFRICON 2017 (2017) 1436–1441.

[11] 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.

[12] 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.

[13] 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.

[14] F.-l. Optimization, Optimization over networks, 7th Edition, Los Angeles, 2011.

[15] J. Xie, C.-C. Liu, Multi-agent systems and their applications, Journal of International Council on Electrical Engineering 7 (1) (2017) 188–197.

[16] A. Ma, A. Nassehi, C. Snider, Anarchic manufacturing, International Journal of Production Research 57 (8) (2019) 2514–2530.

[17] M. Bortolini, F. G. Galizia, C. Mora, Reconfigurable manufacturing systems: Literature review and research trend, Journal of Manufacturing Systems 49 (July 2017) (2018) 93–106.

[18] H. Meissner, R. Ilsen, J. C. Aurich, Analysis of Control Architectures in the Context of Industry 4.0, Procedia CIRP 62 (2017) 165–169.

[19] 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.

[20] B. Esmaeilian, S. Behdad, B. Wang, The evolution and future of manufacturing: A review, Journal of Manufacturing Systems 39 (2016) 79–100.

[21] 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.

[22] N. Kokash, An introduction to heuristic algorithms, Department of Informatics and Telecommunications (August) (2005) 1–8.

[23] S. Nesmachnow, An overview of metaheuristics: accurate and efficient methods for optimisation, International Journal of Metaheuristics 3 (4) (2014) 320.

[24] A. R. Mosteo, L. Montano, A survey of multi-robot task allocation, 2010, pp. 1–27.

[25] 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.

[26] 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.

[27] T. Vidal, J. Bidot, Dynamic sequencing of tasks in simple temporal networks with uncertainty, CP 2001 Workshop in Constraints and Uncertainty (2001) 1–10.

[28] L. Minglei, W. Hongwei, Q. Chao, A novel HTN planning approach for handling disruption during plan execution, Applied Intelligence 46 (4) (2017) 800–809.

[29] 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.

[30] N. Atay, B. Bayazit, Mixed-Integer Linear Programming Solution to Multi-Robot Task Allocation Problem, Tech. Rep. 314 (2006).

[31] A. Mohammad, O. Saleh, R. A. Abdeen, Occurrences Algorithm for String Searching Based on Brute-force Algorithm, Journal of Computer Science 2 (1) (2006) 82–85.

[32] O. Karasakal, L. Kandiller, N. E. Özdemirel, A branch and bound algorithm for sector allocation of a naval task group, Naval Research Logistics 58 (7) (2011) 655–669.

[33] W.-C. Yeh, An efficient branch-and-bound algorithm for the two-machine bicriteria flowshop scheduling problem, Journal of Manufacturing Systems 20 (2) (2002) 113–123.

[34] B. Coltin, M. Veloso, Mobile robot task allocation in hybrid wireless sensor networks, IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings (2010) 2932–2937.

[35] S. Giordani, M. Lujak, F. Martinelli, A distributed algorithm for the multi-robot task allocation problem (2010).

[36] W. Kmiecik, M. Wojcikowski, L. Koszalka, A. Kasprzak, Task allocation in mesh connected processors with local search meta-heuristic algorithms, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5991 LNAI (PART 2) (2010) 215–224.

[37] E. K. Burke, Y. Bykov, A late acceptance strategy in hill-climbing for exam timetabling problems, in: Proceedings of

the Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), 2008, pp. 1–7.

[38] C. Sarkar, H. S. Paul, A. Pal, A Scalable Multi-Robot Task Allocation Algorithm, in: Proceedings - IEEE International Conference on Robotics and Automation, IEEE, 2018, pp. 5022–5027.

[39] 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.

[40] M. Hamzeei, R. Z. Farahani, H. Rashidi-Bejgan, An exact and a simulated annealing algorithm for simultaneously determining flow path and the location of P/D stations in bidirectional path, Journal of Manufacturing Systems 32 (4) (2013) 648–654.

[41] H. J. Fraire Huacuja, J. J. Gonzalez Barbosa, P. Bouvry, A. A. S. Pineda, J. E. Pecero, An iterative local search algorithm for scheduling precedence-constrained applications on heterogeneous machines (2010).

[42] J. Kratica, A. Savi, V. Filipovi, M. Milanovi, Solving the task assignment problem with a variable neighborhood search, Serdica Journal of Computing 4 (4) (2010) 435–446.

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

[44] B. Sharda, A. Banerjee, Robust manufacturing system design using multi objective genetic algorithms, Petri nets and Bayesian uncertainty representation, Journal of Manufacturing Systems 32 (2) (2013) 315–324.

[45] S. Lee, H. G. Kahng, T. Cheong, S. B. Kim, Iterative two-stage hybrid algorithm for the vehicle lifter location problem in semiconductor manufacturing, Journal of Manufacturing Systems 51 (January) (2019) 106–119.

[46] A. Klyne, K. Merrick, Task Allocation Using Particle Swarm Optimisation and Anomaly Detection to Generate a Dynamic Fitness Function Adam, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9457 (2015) 317–329.

[47] L. Brezocnik, I. Fister Jr., V. Podgorelec, Scrum Task Allocation Based on Particle Swarm Optimization, Springer International Publishing AG, part of Springer Nature 2018 10835 LNCS (2018) 38–49.

[48] L. Zhang, J. Sun, C. Guo, H. Zhang, A Multi-swarm Competitive Algorithm Based on Dynamic Task Allocation Particle Swarm Optimization, Arabian Journal for Science and Engineering 43 (12) (2018) 8255–8274.

[49] C. Guan, Z. Zhang, S. Liu, J. Gong, Multi-objective particle swarm optimization for multi-workshop facility layout problem, Journal of Manufacturing Systems 53 (2019) 32–48.

[50] C. Liu, A. Kroll, Memetic algorithms for optimal task allocation in multi-robot systems for inspection problems with cooperative tasks (2014). doi:10.1007/s00500-014-1274-0.

[51] 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.

[52] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, J. R. Woodward, A classification of hyper-heuristic approaches: Revisited, International Series in Operations Research and Management Science 272 (2010) 453–477.

[53] P. R. Ć, S. M. Ć, Dynamic Programming Approach for the Allocation of Limited Resources, Metalurgia International XVII (11) (2012) 2–5.

[54] H. Liang, F. Kang, A novel task optimal allocation approach based on Contract Net Protocol for Agent-oriented UUV swarm system modeling, Optik 127 (8) (2016) 3928–3933.

[55] K. D. Demedeiros, Task Management Using Token Passing for a Group of Cooperating Unmanned Undersea Vehicles (January) (2011).

[56] A. Viguria, I. Maza, A. Ollero, SET: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets Antidio, IIEEE International Conference on Robotics and Automation (2007) 3339–3344.

[57] H. L. Choi, L. Brunet, J. P. How, Consensus-based decentralized auctions for robust task allocation, IEEE Transactions on Robotics 25 (4) (2009) 912–926.

[58] P. A. Gao, Z. X. Cai, L. L. Yu, Evolutionary computation approach to decentralized multi-robot task allocation, 5th International Conference on Natural Computation, ICNC 2009 5 (2009) 415–419.

[59] M. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, A. Kleywegt, Simple auctions with performance guarantees for multi-robot task allocation, 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566) 1 (2004) 698–705.

[60] 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.

[61] 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.

[62] F. Liu, S. Liang, X. Xian, Multi-robot task allocation based on utility and distributed computing and centralized determination, in: Proceedings of the 2015 27th Chinese Control and Decision Conference, CCDC 2015, 2015, pp. 3259–3264.

[63] E. Nunes, M. Gini, Multi-Robot Auctions for Allocation of Tasks with Temporal Constraints, AAAI Conference on Artificial Intelligence (2000) 2110–2116.

[64] L. Luo, N. Chakraborty, K. Sycara, Distributed Algorithms for Multirobot Task Assignment with Task Deadline Constraints, IEEE Transactions on Automation Science and Engineering 12 (3) (2015) 876–888.

[65] M. McIntire, E. Nunes, M. Gini, Iterated multi-robot auctions for precedence-constrained task scheduling, in: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, 2016, pp. 1078–1086.

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

[67] D.-H. Lee, Resource-based task allocation for multi-robot systems, Robotics and Autonomous Systems 103 (2018) 151–161.

[68] C. Sung, N. Ayanian, D. Rus, Improving the performance of multi-robot systems by task switching, Proceedings - IEEE International Conference on Robotics and Automation (2013) 2999–3006.

[69] S. Trigui, A. Koubaa, O. Cheikhrouhou, H. Youssef, H. Bennaceur, M. F. Sriti, Y. Javed, A distributed market-based algorithm for the multi-robot assignment problem, Procedia Computer Science 32 (2014) 1108–1114.

[70] M. P. Fanti, M. Franceschelli, A. M. Mangini, G. Pedroncelli, W. Ukovich, Discrete consensus in networks with constrained capacity, Proceedings of the IEEE Conference on Decision and Control (2013) 2012–2017.

[71] N.˜Boucké, D.˜Weyns, T.˜Holvoet, K.˜Mertens, Decentralized allocation of tasks with delayed commnencement, 2nd European Workshop on Multi-Agent Systems, EUMAS (2004).

[72] M. B. Dias, TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments, Tech. rep. (2004).

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

[74] D. Weyns, N. Bouck, K. Schelfthout, T. Holvoet, Dyncnet: A protocol for flexible task assignment applied in an AGV transportation system, CEUR Workshop Proceedings 223 (January) (2006).

[75] A. Viguria, I. Maza, A. Ollero, S+T: An algorithm for distributed multirobot task allocation based on services for improving robot cooperation, Proceedings - IEEE International Conference on Robotics and Automation (2008) 3163–3168.

[76] Q. C. Ye, Y. Zhang, R. Dekker, Fair task allocation in transportation, Omega (United Kingdom) 68 (2017) 1–16.

[77] R. Erol, C. Sahin, A. Baykasoglu, V. Kaplanoglu, A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems, Applied Soft Computing Journal 12 (6) (2012) 1720–1732.

[78] W. Pereira, G. S. Bastos, Robotics, Brazilian Conference on Robotics Latin American Robotics Symposium 619 (2016) 210–227.

[79] W. P. N. D. Reis, G. S. Bastos, Multi-Robot Task Allocation Approach Using ROS, Proceedings - 12th LARS Latin American Robotics Symposium and 3rd SBR Brazilian Robotics Symposium, LARS-SBR 2015 - Part of the Robotics Conferences 2015 (2016) 163–168.

[80] T. Dahl, M. Mataric, G. Sukhatme, Multi-robot task-allocation through vacancy chains, IEEE International Conference on Robotics & Automation 2 (2003) 12–17.

[81] D. Weyns, N. Boucké, T. Holvoet, Gradient field-based task assignment in an AGV transportation system, Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems - AAMAS '06 (2006).

[82] G. J. Cawood, I. A. Gorlach, Navigation and locomotion of a low-cost Automated Guided Cart, Proceedings of the 2015 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference, PRASA-RobMech 2015 (2015) 83–88.

[83] H. M. Barbera, J. P. C. Quinonero, M. A. Z. Izquierdo, A. G. Skarmeta, i-Fork: a flexible AGV system using topological and grid maps, in: IEEE International Conference on Robotics & Automation, IEEE, 2003, pp. 2147–2152.

[84] S. Lu, C. Xu, R. Y. Zhong, L. Wang, A RFID-enabled positioning system in automated guided vehicle for smart factories, Journal of Manufacturing Systems 44 (2017) 179–190.

[85] J. Szpytko, P. Hyla, Automated Guided Vehicles Navigating Problem In Container Terminal, Logistics and Transport 2 (13) (2011) 107–116.

[86] J. Song, Electromagnetic induction sensor of navigation system for spraying robot, Advances in Intelligent and Soft Computing 112 (2011) 175–181.

[87] C. Feledy, S. Luttenberger, A State of the Art Map of the AGVS Technology and a Guideline for How and Where to Use It, Tech. rep., University of Lund, Lund (2017).

[88] S.-y. Lee, H.-w. Yang, Robotics and Computer-Integrated Manufacturing Navigation of automated guided vehicles using magnet spot guidance method, Robotics and Computer Integrated Manufacturing 28 (3) (2012) 425–436.

[89] U. Andersson, Laser Navigation System for Automatic Guided Vehicles, Tech. rep., Lulea (2013).

[90] H. Durrant-whyte, T. Bailey, Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms, IEEE Robotics & Automation Magazine 13 (2) (2006) 99–110.

[91] A. M. Santana, K. R. Aires, R. M. Veras, A. A. Medeiros, An approach for 2D visual occupancy grid map using monocular vision, Electronic Notes in Theoretical Computer Science 281 (2011) 175–191.

[92] W. Elmenreich, An introduction to sensor fusion, Vienna University of Technology, Austria (2002) 1–28.

[93] G. Welch, G. Bishop, An Introduction to the Kalman Filter, in: Siggraph Course. 8, 2006, pp. 1–16.

[94] S. W. Yoon, S. B. Park, J. S. Kim, Kalman filter sensor fusion for Mecanum wheeled automated guided vehicle localization, Journal of Sensors 2015 (2015) 1–8.

[95] E. Cardarelli, V. Digani, L. Sabattini, C. Secchi, C. Fantuzzi, Cooperative cloud robotics architecture for the coordination of multi-AGV systems in industrial warehouses, Mechatronics 45 (2017) 1–13.

[96] V. Kunchev, L. Jain, V. Ivancevic, A. Finn, Path Planning and Obstacle Avoidance for Autonomous Mobile Robots: A Review, in: KES2006 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems, 2006, pp. 537–544.

[97] S. G. Anavatti, S. L. Francis, M. Garratt, Path-planning modules for Autonomous Vehicles: Current status and challenges, ICAMIMIA 2015 - International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation, Proceeding - In conjunction with Industrial Mechatronics and Automation Exhibition, IMAE (2016) 205–214.

[98] Ahmad Abbadi, Vaclav Prenosil, Safe Path Planning Using Cell Decomposition Approximation, International Conference Distance Learning, Simulation and Communication (May) (2015).

[99] N. Tran, D. T. Nguyen, D. L. Vu, N. V. Truong, Global path planning for autonomous robots using modified visibility-graph, 2013 International Conference on Control, Automation and Information Sciences, ICCAIS 2013 (2013) 317–321.

[100] S. M. Sabra, M. M. Soliman, The prevalence of impacted mandibular wisdom with associated physical signs and microbial infections among under graduate girls at Taif University, KSA, World Applied Sciences Journal 21 (1) (2013) 21–29.

[101] M. T. Rantanen, M. Juhola, A configuration deactivation algorithm for boosting probabilistic roadmap planning of robots, International Journal of Automation and Computing 9 (2) (2012) 155–164.

[102] R. Seif, M. A. Oskoei, Mobile Robot Path Planning by RRT* in Dynamic Environments, International Journal of Intelligent Systems and Applications 7 (5) (2015) 24–30.

[103] J. A. Herrera Ortiz, K. Rodrguez-Vázquez, M. A. Padilla Castaeda, F. Arámbula Coso, Autonomous robot navigation based on the evolutionary multi-objective optimization of potential fields, Engineering Optimization 45 (1) (2013) 19–43.

[104] D. H. Kim, N. T. Hai, W. Y. Joe, A Guide to Selecting Path Planning Algorithm for Automated Guided Vehicle (AGV), Lecture Notes in Electrical Engineering 465 (2018) 587–596.

[105] K. M. R. L. Moorthy, W. H. Guan, Deadlock Prediction and Avoidance in an AGV System, Ph.D. thesis (2000).

[106] S. C. Srivastava, A. K. Choudhary, Development of an intelligent agent-based AGV controller for a flexible manufacturing system, International Journal Advanced Manufacturing Technology 36 (2008) 780–797.

[107] D. K. Liu, A. K. Kulatunga, Simultaneous planning and scheduling for multi-autonomous vehicles, Studies in Computational Intelligence 49 (2007) 437–464.

[108] 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.

[109] V. Digani, F. Caramaschi, L. Sabattini, C. Secchi, C. Fantuzzi, Obstacle avoidance for industrial AGVs, Proceedings - 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing, ICCP 2014 (2014) 227–232.

[110] A. Winkler, J. Suchý, Dynamic collision avoidance of industrial cooperating robots using virtual force fields, in: IFAC Proceedings Volumes (IFAC-PapersOnline), Vol. 45, 2012, pp. 265–270.

[111] G. Sahin, M. Balcilar, E. Uslu, S. Yavuz, M. F. Amasyali, Obstacle avoidance with Vector Field Histogram algorithm for search and rescue robots, IEEE 22nd Signal Processing and Communications Applications Conference (Siu) (2014) 766–769.

[112] D. An, H. Wang, VPH: A new laser radar based obstacle avoidance method for intelligent mobile robots, Proceedings of the World Congress on Intelligent Control and Automation (WCICA) 5 (2004) 4681–4685.

[113] Y. Kang, D. A. De Lima, A. C. Victorino, An approach of human driving behavior correction based on Dynamic Window Approach, IEEE Intelligent Vehicles Symposium, Proceedings (Iv) (2014) 304–309.

[114] S. J. Guy, M. Lin, D. Manocha, Reciprocal n -body Collision Avoidance, in: Robotics Research, star 70 Edition, Springer, Berlin Heidelberg, 2011, pp. 1–16.

[115] Y. F. Chen, M. Liu, M. Everett, J. P. How, Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning, Proceedings - IEEE International Conference on Robotics and Automation (2017) 285–292.

[116] W. Hu, Y. Zhu, J. Lei, The Detection and Prevention of Deadlock in Petri Nets, Physics Procedia 22 (2011) 656–659.

[117] G. Mejía, N. G. Odrey, An approach using petri nets and im-

proved heuristic search for manufacturing system scheduling, Journal of Manufacturing Systems 24 (2) (2005) 79–92.

[118] 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) 61–66.

[119] Z. Li, N. Wu, M. Zhou, Deadlock control of automated manufacturing systems based on petri nets-a literature review, IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews 42 (4) (2012) 437–462.

[120] 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.

[121] J. W. Yoo, E. S. Sim, C. Cao, J. W. Park, An algorithm for deadlock avoidance in an AGV System, International Journal of Advanced Manufacturing Technology 26 (5-6) (2005) 659–668.

[122] M. Lehmann, M. Grunow, H. O. Günther, Deadlock handling for real-time control of AGVs at automated container terminals, Container Terminals and Cargo Systems: Design, Operations Management, and Logistics Control Issues 657 (2007) 215–241.

[123] 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.

[124] 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.

[125] 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.

[126] D. Marino, A. Fagiolini, L. Pallottino, Distributed Collision-free Protocol for AGVs in Industrial Environments (2011).

[127] 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.

[128] Y. C. Ho, T. W. Liao, Zone design and control for vehicle collision prevention and load balancing in a zone control AGV system, Computers and Industrial Engineering 56 (1) (2009) 417–432.

[129] Y. C. Ho, Dynamic-zone strategy for vehicle-collision prevention and load balancing in an AGV system with a single-loop guide path, Computers in Industry 42 (2) (2000) 159–176.

[130] M. P. Fanti, Event-based controller to avoid deadlock and collisions in zone-control AGVS, International Journal of Production Research 40 (6) (2002) 1453–1478.

[131] R. W. McHaney, Modelling battery constraints in discrete event automated guided vehicle simulations, International Journal of Production Research 33 (11) (1995) 3023–3040.

[132] Q. S. Kabir, Y. Suzuki, Comparative analysis of different routing heuristics for the battery management of automated guided vehicles, International Journal of Production Research 57 (2) (2018) 624–641.

[133] M. Ebben, Logistic Control In Automated Transportation Networks, Ph.D. thesis (2001).

[134] Q. S. Kabir, Y. Suzuki, Increasing manufacturing flexibility through battery management of automated guided vehicles, Computers and Industrial Engineering 117 (January) (2018) 225–236.

[135] T. Kawakami, S. Takata, Battery Life Cycle Management for Automatic Guided Vehicle Systems, Design for Innovative Value Towards a Sustainable Society (2012) 403–408.

[136] M. De Ryck, M. Versteyhe, K. Shariatmadar, Resource Management in Decentralized Industrial Automated Guided Vehicle Systems, Journal of Manufacturing Systems (2019).

[137] K. Shariatmadar, K. Driesen, M. De Ryck, F. Debrouwere, M. Versteyhe, Linear programming under $\epsilon$-contamination uncertainty, in: International Conference Computational and Mathematical Methods in Science and Engineering, 2019.

[138] K. Shariatmadar, K. Driesen, M. De Ryck, F. Debrouwere, M. Versteyhe, Optimization under e-contamination uncertainty, Computational and Mathematical Methods (2019) 1–15.