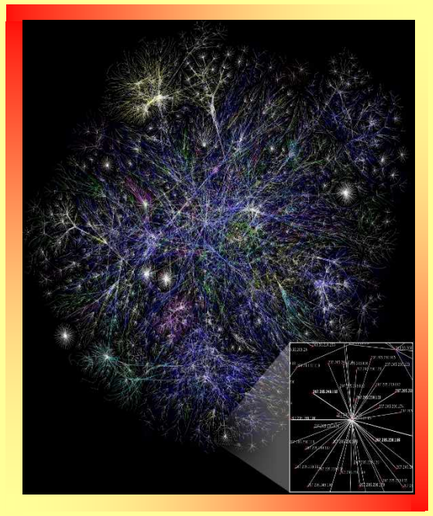
UNIVERSITÀ
DEGLI STUDI DI
SALERNO

Ministero dell'Istruzione,
dell'Università e della Ricerca

Tesi di Dottorato/Ph.D. Thesis

An Auction-based Approach to Control Automated Warehouses using Smart Vehicles

Emiliano Di Marino



Supervisor: **Prof. Francesco Basile**

Ph.D. Program Director: **Prof. Pasquale Chiacchio**

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata
Dipartimento di Informatica

Ciclo 32 – AA 2018/2019

Dottorato di Ricerca in Informatica e Ingegneria dell'Informazione

Emiliano Di Marino – An Auction-based Approach to Control Automated Warehouses using Smart Vehicles

Università degli Studi di Salerno

Dottorato di Ricerca in Informatica e Ingegneria dell'Informazione
Ciclo 32 – a.a 2018/2019

TESI DI DOTTORATO / PH.D. THESIS

An Auction-based Approach to Control Automated Warehouses using Smart Vehicles

EMILIANO DI MARINO

SUPERVISOR: **PROF. FRANCESCO BASILE**

PHD PROGRAM DIRECTOR: **PROF. PASQUALE CHIACCHIO**

Dipartimento di Ingegneria dell'Informazione ed Elettrica
e Matematica Applicata
Dipartimento di Informatica

Contents

A Mamma e Papà,
per i loro immensi sacrifici.

A Federico,
per il suo instancabile sostegno.

Ad Annamaria,
per essere stata forte nei momenti difficili,
per spronarmi a non lasciare mai nulla d'intentato.

A Martina,
per avermi ascoltato, sempre.

Agli amici,
perché senza di loro tutto è più difficile.

- A un nuovo capitolo della mia vita. -

Chapter 1

Introduction

Warehouses play a critical role in the flow of goods from manufacturers to consumers and their basic functions (i.e. receiving, storage, order picking and shipping) are essential components in any supply chain [1]. According to [2], in 2007, more than 80% of all Western European warehouses still operated according to the traditional picker-to-parts principle: human operators (i.e. pickers) continuously visit storage racks consisting of shelves on which the required Stocking Units (SUs) are stored and, after collecting a certain number of items therein, return to a central depot. The main drawback of these approaches is, of course, the unproductive walk of the picker as well as finding the most efficient route to collect all the items to satisfy a certain number of orders. A schematic representation of such a set-up is shown in Fig. 1.1 where the storage racks and the central depot are depicted with green and yellow rectangles respectively. Nowadays, such systems are slowly disappearing and are being replaced by those characterized by a certain degree of automation (e.g. automated picking/storage workstations, automated cranes) or by certain organizational adaptions (e.g. mixed-shelves storage, dynamic order processing, and batching, zoning and sorting systems). Thus, modern Automated Warehouse Systems (AWSs) are capable of meeting a large number of time-critical orders: they are specifically designed to meet the needs of (online) retailers to serve final customers (espe-

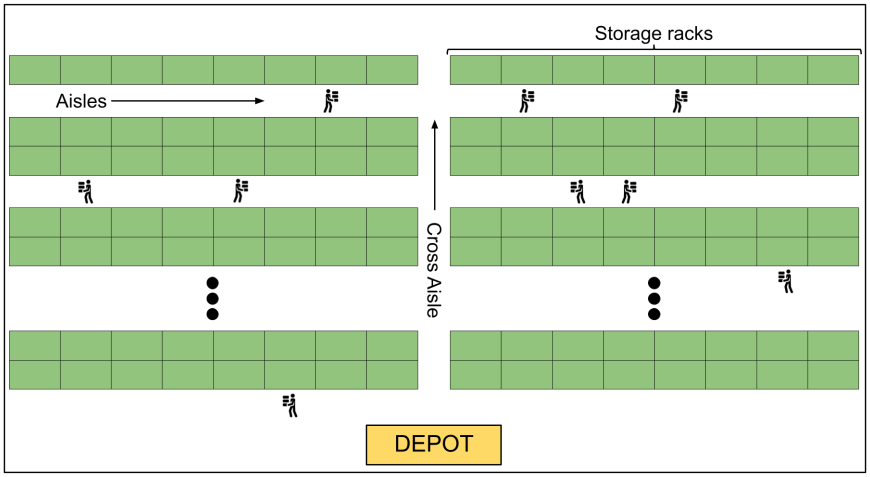


Figure 1.1: Example scheme of a warehouse with a picker-to-part set-up: the storage racks and the central depot are depicted with green and yellow rectangles respectively.

cially) in the business-to-consumer segment [3].

Most of the modern AWSs are characterized by a big Interface System (IS) composed by vehicles which can move SUs along a mono-dimensional guide-path [4]. They perform picking actions (from the output bays of the storage area and/or from the output bays of the picking area) and deposit actions (into the input bays of the storage area and/or into input bays of the picking area). The picking area is a human-intensive space where workers collect items from the SUs in a certain quantity to satisfy customer orders. The storage area is generally composed by several crane-served racks in which SUs are stored and retrieved when necessary. The first and the second area interface with the IS through input/output buffers. With this kind of set-up, modern AWSs have a high picking performance and this makes them suitable to accomodate orders with tight deadlines [1]. Also, small order sizes and a large assortment seem unproblematic, as long as there is enough space in the storage area [1].

A set of tasks (missions) is given as input to this kind of systems. A task requires that a SU must be moved from the output

(input) buffer of a picking (deposit) bay to the input (output) buffer of a deposit (picking) bay or viceversa. Thus, vehicles autonomously pick (deposit) a SUs from a predefined output (input) buffer while human operators remain in the storage and/or picking area.

The efficiency and the effectiveness of any AWS is largely determined by the adopted planning and control strategies as well as by the characteristics of the storage area, the picking area and the IS. In general, the design of the latter subsystems ranges from functional description, through a technical specification, to equipment selection and layout determination. At each stage, the performance criteria (e.g. costs, throughput, storage capacity) must be satisfied. As a result, designing an AWS is a very time-consuming task and compromises must be made between often conflicting objectives [5]. In this work, the control of the IS is addressed since the control of the storage area and of the picking area has been exhaustively investigated in the literature. Indeed, regarding the storage area, quite a few automated compact storage systems have been developed and studied in recent years. Examples are shuttle-based deep lane storage systems [6], the live-cube system [7], and puzzle-based storage systems [8]. A detailed overview on compact storage is provided in [9]. These systems efficiently exploit the space by storing the products closest to each other without providing direct access to each individual item: with intelligent planning and automation approaches they try to ensure sufficiently fast retrivial times. However, such compact storage systems are not well suited for the tight delivery schedules [1]. An example of a compact storage system is shown in Fig. 1.2. In constrast, fully automated A-Frame systems described in [10] are well suited to fulfill tight deadlines as well as small order sizes [1]. The main drawback is related to the constraints on the product shape, size, and packaging. In addition, an excessive number of A-Frame modules is required for large assortments leading to high investment costs. Other types of systems have also been developed to handle specific product types constrained by their weight and/or shape [1]. Referring to the picking area, it is typically characterized by bays

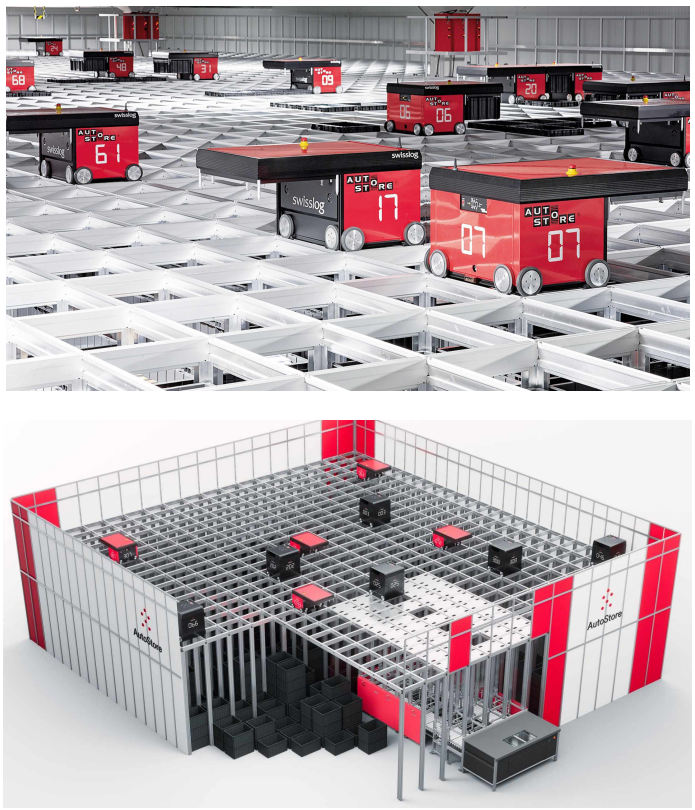


Figure 1.2: A compact storage system by Swisslog that uses robots and bins to quickly process small parts orders.

each one consisting of a local storage system, an intermediate conveyor belt and a specific place where a worker collects all the items that need to be shipped to satisfy a certain customer order as the one depicted in Fig. 1.3. They have high performance and seem well suited to meet tight deadlines. In addition, large assortments and small-size orders do not appear to be problematic. The main drawback is the scalability: adding/removing bays is expensive [1]. Most of researches focus on the storage system characterized by carousels or, automated cranes and lift and shuttle systems [1]. Other works deal with simulation approaches and queuing models to evaluate different layout configurations [11].

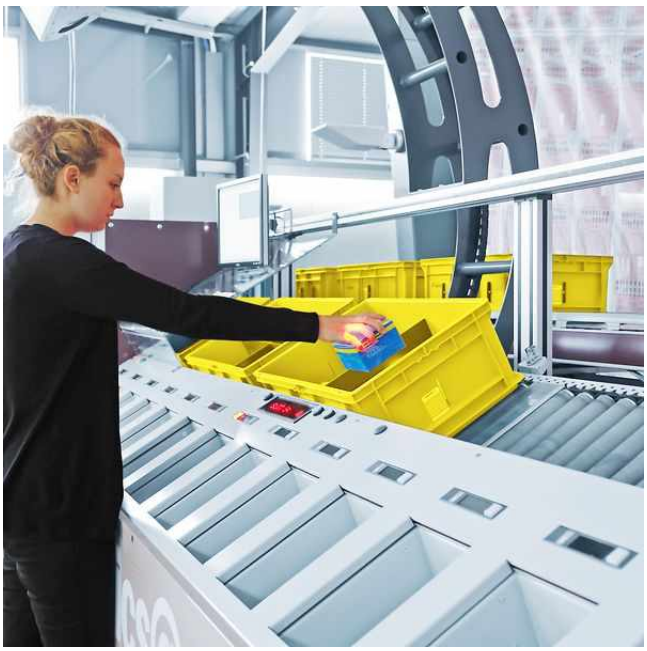


Figure 1.3: The ergonomic picking system from SSI SCHAEFER which allows up to 1000 picks per hour.

1.1 Position of the thesis

A relevant control problem related to the IS consists in assigning each available vehicle to a mission. This can be divided into two sub-problems: the first concerns how a set of missions is assigned to a set of vehicles; the second regards how the behavior of the fleet of vehicles is coordinated in order to perform missions efficiently and reliably. Because the missions allocation problem is a dynamic decision problem that varies in time with phenomena, the problem should be solved iteratively over time [12]. Thus, it is easy to understand that the mission allocation problem becomes more complex to tackle and the requirements of the logistic domain affects the complexity of such a problem [13]. In this work, the first sub-problem is extensively studied while the second is not covered: as discussed below, it is assumed that vehicles can

automatically regulate their velocity to avoid collisions and deadlock prevention policies can be exploited using existing algorithms present in literature, as the one of [14]. Thus, the addressed control problem is to establish "*Who* has to do *What* and in *Which Way*" in order to optimize a predefined global objective function. In detail, it is associated to:

1. the identification of which vehicle (*Who*) must perform a certain mission (*What*);

2. the identification of the best strategy to accomplish such a mission (*Which Way*).

Both issuess must be solved by optimizing a predefined global objective function: in this work, the makespan, defined as the time required to the IS to complete a set of missions, is considered. Nowadays, it seems natural to assume that vehicles are equipped with high-performance, low-power electronic components that provide communication capabilities and processing power [15]. Moreover, as assumed in this work, each vehicle is often able detect obstacles (e.g. other vehicles) on their path and can consequently modify their velocity to avoid collisions. Indeed, as discussed in [16] vehicles are generally able to detect obstacles using simple low-cost 2D laser scanners and can use 3D information coming from LIDAR sensors and/or 3D time-of-flight cameras. In this way, vehicles can also safely operate in structured environments, stopping the execution of a certain task if a potential risk is detected. Thus, they are able to understand their surroundings, recognize lane markings, able to work on their own and interact with each other exchanging messages. These vehicles are also known as "smart vehicles" and will be simply referred as vehicles in the following, when the context is clear. In Fig. 1.4 an example of two smart vehicles that move in an AWS is reported.

In this work, an auction-based control architecture to address the IS mission assignment is proposed. It is worth remarking that, although the approach is conceived for AWSs, it can be applied to many logistic systems because the considered control problem is absolutely general in the logistic context.

(a) Movexx vehicle

(b) Indeva Tugger vehicle

Figure 1.4: Example of two smart vehicles that use their sensors to navigate along a mono-dimensional guide-path without any human intervention in two different AWSs.

1.1.1 Motivations behind auction-based control architectures

Auction-based approaches are mostly used in the robotic community to address the control of multi-robot systems. They rely on three important entities: sellers, who sell items; buyers (bidders), who buy them; an auctioneer, who mediates between the two previous players and assigns the items on the basis of the bid of each buyer. The negotiation between the previous entities happens via auctions: they have been exploited in societies throughout history to distribute resources among individuals and groups [17]. In general, any protocol that allows an entity to state an interest in one

or more resources and to obtain them is considered an auction. As a result, different types of auctions have been devised and can be divided into two main categories namely simple-good auctions and combinatorial auctions respectively [17]. Fig. 1.5 shows the taxonomy for different auctions types. In simple-good auctions, buyers can bid on individual items while, in combinatorial auctions, they can bid on a (discrete) combination of them. In addition, the first one can be distinguished in open-cry and closed-cry auctions: in the first types, the identity of all bidders is made known during the auction, while, in the second types, this characteristic is not respected. Open-cry auctions are also divided into:

- *English auctions* (also referred as open-cry ascending auctions), where bidders submit progressively larger bids with the aim of overcoming each other and the highest bidder's bid wins;
- *Dutch auctions* (also referred as open-cry descending auctions), where at the beginning the auctioneer proposes a high starting price for the item to be sold and than starts to lower it gradually until one of the bidders makes a bid, winning the auction.

Closed-cry auctions are divided into:

- *Sealed-bid auctions* (also referred as first-price sealed-bid auctions), where all bidders simultaneously submit sealed bids to the auctioneer (so that no bidder knows how much the other auction participants have bid) and the highest bid wins. Then, the winner, pays its bid;
- *Vickrey auctions* (also referred as second-price sealed-bid auctions), that maintains the same characteristics of the previous ones except for the fact that the price paid at the end of the auction is equal to the second-highest bidder's bid.

Concerning the combinatorial auctions, they allow buyers to compete for the entire batch of items or for a subset of such a

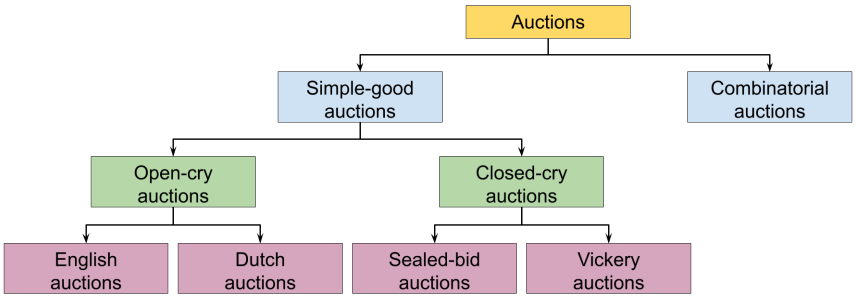


Figure 1.5: Auctions types.

batch. In the second case, not easily solvable conflict problems arise: multiple buyers can bid on the same items and may also offer the same price.

Regardless of the type of auction used, most auction-based methods share a number of basic elements [18]:

1. all or some of the entities involved in the task allocation process compete for a task (which can be decomposed into subtasks) as individuals or in groups;

2. a global objective function is defined to meet certain system performance requirements;

3. an individual/group utility (or cost) function is specified for each involved entity or groups of entities and quantifies it-s/their interests in the task to perform;

4. a mapping is defined between the global objective function and the individual/group utility.

All these aspects reflect the nature of the presented control problem and therefore auction-based approaches can also be applied in the logistic context. Thus, in AWSs, simple-good auctions can be simply exploited to efficiently allocate missions to vehicles. In addition, combinatorial auctions and those that may include the use of the "reserve price" to set a minimum/maximum price at which the items must be sold are very attractive as they provide

significant advantages in the above mentioned application domain: the former, permit to reduce the number of auctions required to allocate a large amount of items, and, the latter, allow to establish only a subset of buyers to communicate with (i.e. buyers who do not have enough resources for the item do not propose any bid).

To the best of author's knowledge, this work is the first one that applies an auction-based control architecture in AWSs for the control of vehicles and it is motivated by the optimistic results obtained by previous researchers of the robotic community. Indeed, Motivated by the optimistic results obtained by previous researchers for the control of multi-robot systems, in this work an auction-based control architecture is explicitly designed for the AWS context bringing to the whole system:

1. *modularity*, that allows to take into account the different characteristics of each vehicle in the mission allocation process (i.e. designing an appropriate utility function [17]);

2. *real-time-capabilities*, that lead to establish "*Who* has to do *What* and in *Which Way*" in a very fast and flexible manner, since vehicles are able to quickly perform on-board calculations;

3. *robustness*, that permits to set-up a fault tolerant system, since each vehicle is able to make its own decisions;

4. *services orientation capabilities*, that enable the use of a plug-and-operate approach;

5. *scalability*, that permits to easily scale to applications with a large number of vehicles because the computational effort results to be distributed.

Also notice that:

- the real-time-capabilities allow removing the central control unit of typical hierarchical and centralized approaches;

- the services orientation paradigm makes it possible to introduce new vehicles into the system without the need of a physical configuration: services help to abstract the hardware layer of the vehicles and permit to exhibit the capability of each of them in a very flexible way.

It is also important to mention that auction-based control architectures (which implement the above described features) are aligned with the recommended requirements and needs of the "Industry 4.0" paradigm.

The most important drawbacks of auction-based approaches are [17]:

1. the lack of formal guidelines to design and develop such control systems;

2. the computation/communication requirements to perform auctions.

As also explained in details in Sec. 1.2, to overcome the first issue and also to face with the problem of the system performance evaluation and analysis, a formal model is used together with heuristic based control laws. Such an approach allows savings costs and time during the system design phase that are two important factors to take into account when complex systems must be deployed. Furthermore, as will additionally be shown in Sec. 5.2, a design methodology can be also devised for AWSs.

Concerning computational and communication requirements, they become critical issues when the number of involved entities in the system is very large and each entity is not able to perform any type of calculation. However, in logistic environments it is possible to observe that:

- The number of employed vehicles is small and limited [19] to few dozen: small plants include up to 10 vehicles, medium plants include 10 to 30 vehicles, big plants include more than 30 vehicles (and in practice less than 70 vehicles). On the other hand, in robotic community there is often the need to

coordinate a large number of robots in static and/or dynamic environments as detailed, for example, in [20].

- Vehicles can move SUs only among a finite and small set of routes since the industrial environments are well structured.
- Vehicles have on-board computing capabilities.

This leads to the conclusion that computational and communication requirements must not be considered to be a disadvantage. Furthermore, with the use of heuristics, vehicles need only few computational power to make decisions during the mission allocation process and with the adopted auctions, the employed control architecture permits to obtain efficient solutions. Notice that with the use of auction-based approaches network delays should be taken into account: in the considered scenario, where the entities involved in the communications are in the order of several dozen units, network delays can range from few milliseconds to some seconds [21, 22, 23].

For the purpose of this work, simple-good auctions and combinatorial auctions are divided into multi-round and single-round auctions. In the first group fall all those that employ more than one round to allocate an item; in the second fall all the auctions where an item is one-shot assigned. In Fig. 1.6 the proposed taxonomy is represented. Consequently, the ascending/descending auctions belong to the first group while the sealed-bid auctions and the Vickery auctions are in the second. Notice that combinatorial auctions can be performed using one or more rounds and, as a result, fall into the intersection of the two previously defined groups. In addition, in both groups other variants are possible.

Multi-round auctions can be easily used in all the contexts in which:

1. the response time (i.e. the total time required to assign a mission to a vehicle) is not a crucial factor to find the optimal allocation;
2. bids depend from a pre-defined studied strategy (i.e. an action plan designed to achieve a particular objective).

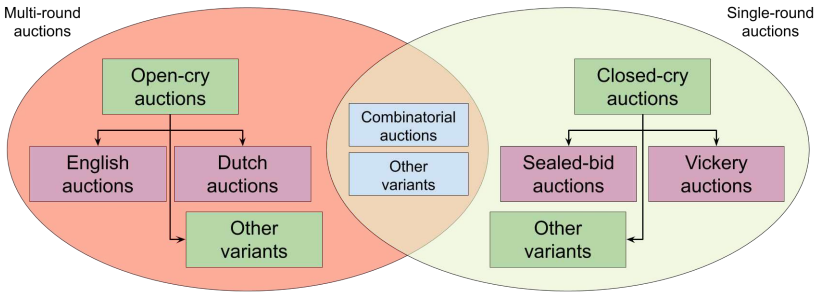


Figure 1.6: Multi-round and single-round auctions.

Concerning the first point, in AWSs the response time to assign a mission makes unattractive the use of multi-round auctions due to their demanding computation/communication requirements. Indeed, assuming N_r and N_v as the number of rounds needed to assign a mission and the number of vehicles respectively, $N_r \times N_v$ messages among all the entities must be exchanged to terminate the allocation process [24] and, the auctioneer, must evaluate the same amount of bids. In contrast, in single-round auctions, the item is one-shot assigned: only N_v messages are exchanged and just N_v bids must be considered by the auctioneer. Concerning the second point, each vehicle does not follow a predefined action plan to meet a particular objective: its bid is designed to express its ability in the mission to perform with respect to the path from its actual position to some specific locations related to each mission. Thus, considering the on-board capabilities of the vehicles, in this work only variants of one-shot k-price sealed-bid auctions are proposed and analysed. The parameter k (as extensively detailed in Sec. 3) is aimed at choosing a vehicle which, at the end of an auction, does not exceed the location where a SU is to be withdrawn.

Features	**Used approaches in AWSs**		
	Centralized [26] [27] [28] [29] [19] [30]	**Decentralized** [31] [32] [33] [34]	**Auction-based**
Optimality	The global state of the system is used to achieve the optimal solution but at the expense of a high computational cost.	Distributed computation and the level of autonomy of the vehicles allow to obtain good sub-optimal solutions reducing the computational complexity.	Auctions permit to achieve good sub-optimal solutions: bids are cost-effective methods to evaluate the ability of each vehicle in performing a mission.
Robustness	Not guaranteed because the central control unit is the single point of failure of the the whole system.	Robustness w.r.t. several types of malfunctions is limited by the distributed computation requirements of these approaches.	The level of autonomy of the vehicles together with auctions permits to set-up fully fault tolerant systems.
Scalability	As the size of the vehicles increases, the computational effort becomes intractable in practice.	A large number of vehicle can be handled since, each of them, is able to make its own decisions.	Auctions allow to easily set-up efficient scalable systems w.r.t. the number of vehicles and missions at a reduced computational cost.

Table 1.1: Key differences between the proposed auction-based approach and the existing methods for controlling AWSs w.r.t. the optimality, robustness and scalability.

1.2 Contribution and adopted methodology

In addition to the choice of the adoption of an auction-based control architecture, that is the main contribution of this work, control algorithms based on heuristics are proposed. Although it could be possible to describe the mission allocation problem as a large global optimization problem, it is impractical to do so for a variety of reasons. First, the mission allocation decisions must be made in real time and there are few windows in which an offline computation can be made (since there is also the need to constantly reprioritizing missions to match changing schedules) [41]. Second, the problem description is quite large and may include a huge amount of orders and exponential missions to vehicles assignment choices. Third, the optimal solution also depends on the actual paths and interactions of the vehicles, which is dynamic. Consequently, instead of attempting global optimization, the proposed approach allows to take decisions on the fly using an heuristics-based approach. To this end, a real AWSs has been considered and analyzed following the several stages described in [25]. The first one concerns the acquisition of information, which consists of identifying the different parameters influencing the system. The second, regards the development of a reliable model that must integrate all the identified parameters. The third stage concerns the evaluation and analysis of the system performance using the developed model (that must be simple and easy to use).

It is very important to stress that the development of a reliable model is a crucial aspect to obtain very accurate results. Since the modelling of an AWS can be performed at various levels of complexity, a very important step is to choose the level of modelling (that can range from very aggregate to very detailed) according to the requirements for system analysis and control [26]. It is evident that the use of a detailed formal model is necessary to support the design steps and it would be preferable to use a unique formalism to avoid translations between different modelling domains. Thus, following the approach described in [25], in order to show the ef-

fectiveness of the auction-based control architecture in the logistic context, in this work is presented:

- a detailed highly modular and compact hybrid model based on Petri Nets (PNs) that permits to represent the continuous and discrete dynamics that are both relevant in an IS as discussed in [4]. This is essential to simulate the behavior of the system under different heuristic strategies;

- an automated model synthesis procedure that, on the basis of high level parameters of the IS, automatically generates the correspondent hybrid model that satisfies the system constraints;

- a design methodology to compensate the absence of formal guidelines for designing and developing auction-based control architectures.

Concerning the first point, the use of a PN model is motivated by the fact that such a formalism (together with its variants) has been successfully used to model and analyze many Discrete Event Systems (DESs). In detail, a coloured hybrid PN model is desiged since it is possible to use tools operating on these nets, as the one presented in [14], to execute model simulation and analysis and to approach also the collision prediction and deadlock prevention in these systems. In [14] a methodology is presented to devise a deadlock prevention policy that acts directly on the PNs devised from the coloured hybrid PN model. Deadlock prevention and collision prevention are out of the scope of this work, but, the model obtained applying the presented approach, is a good starting to exploit existing algorithms for PN models. Also notice that, even if it is theoretically possible to exploit the analytical model of the system, it results very difficult to compute and complicated to deal with: for the purpose of this work, the DES framework is well suited for the evaluation of the system performance. The use of this framework is also motivated by the fact that modern robot manufacturers deliver smart vehicle as black-box: they

design their vehicles, chargers, software and services to work together as an integrated system. Thus, their behaviors and their interactions can be easily modeled using hybrid models.

As for the automatic synthesis procedure, it helps to generate specific instances of ISs from specific high-level parameters and it is a safe way to test and explore different "what-if" scenarios. This permits to easily evaluate what happens when some resources (e.g. vehicles, routes) are added/removed to/from IS. It is also an important tool to quickly and effectively manage planned and unplanned situations in which decisions must be taken immediately.

As for the third contribution, a design methodology to compensate the absence of formal guidelines for designing and developing auction-based control architectures is also presented. This is accomplished on the basis of a detailed simulation study, where a set of fully dimensionless parameters common to any AWS is considered: they can be set to starting values guaranteeing acceptable performance, and then, using a more accurate simulation study on the specific case, a finest tuning can be achieved. It is very important to underline the need for a design methodology for auction-based approaches and, in general, for decentralised systems: since there is no central control unit, unpredictable behaviour can arise that make the acceptance of such solutions difficult. Providing a design methodology for such systems helps to solve complex and dynamic decision-making processes that are distributed, providing effective alternatives to typical hierarchical and centralized control approaches adopted by industries [35].

It is also very important to underline that, in contrast to heuristics, "forward simulation" or look-ahead methods can be adopted. However, such methods are very onerous from a computational point of view: this means that time and costs are constraints so relevant to make this solution not practical. As regards the second point, the following considerations can be done: because the time that a vehicle employs to perform a whole mission is of the order of few minutes, it is possible to simulate the behaviour of some vehicles (in some moments) for a certain time horizon. As detailed in Sec. 3, these methods are used in the employed hierarchical and

centralized control architecture.

To conclude this section, in Tab. 1.1, the key differences between the proposed auction-based approach and the existing methods for controlling AWSs with respect to the optimality, robustness and scalability is reported.

1.3 Relevant Literature

Allocating a set of missions to a fleet of vehicles is a very relevant topic in AWSs and, most of the approaches, rely on centralized control architectures. In the following, the literature review is focused on the mission allocation problem and the coordination of the vehicles in the IS; we do not take care of the low level control of the vehicles or the location assignment problem in the warehouse.

In [30] authors propose a modular and unified modelling framework for heterogeneous automated storage and retrieval systems, comprising both the rail guided vehicles and cranes subsystems. In this work, the activity of the automatic storage and retrieval subsystems is not modelled but it is reduced to a timed transition that models the execution time of a given (generalized) storage and retrieval cycle: this permits to obtain a model of reasonable size for real warehouses.

In [26] authors propose centralized control algorithms for the management of AWSs while introducing a new level in their typical hierarchical and centralized control architecture, named optimizer system. In this work, a modular implementation (based on the auction paradigm) of the suggested control architecture is proposed, in order to achieve all the benefits discussed in Sec. 1.1.1.

In [27] a discrete control oriented model which can be employed for monitoring, scheduling and rescheduling of activities by using simulation for manual-pick warehouses is adopted. On the other hand, in this work, an AWS with a part-to-picker setup is considered, for which a hybrid model to capture all the relevant dynamics of the system, as in [36], is proposed.

In [29] an optimization strategy to coordinate a fleet of Auto-

mated Guided Vehicles (AGVs) traveling on pre-defined roadmaps is presented. In particular, authors extend the their previous works [37, 38]: the objective is to maximize traffic throughput of AGVs navigating in an AWS by minimizing the time AGVs spend negotiating complex traffic patterns to avoid collisions with other AGVs. The coordination problem is formulated as a Quadratic Program where the optimization is performed in a centralized manner.

In [31], two decentralized algorithms are proposed in order to solve the main problems involving the AGV systems control: (i) assigning tasks (missions) to AGVs by a consensus based approach; (ii) routing them by a decentralized control and coordination strategy. The main difference with our approach is the computational power required to be installed on each vehicle. Indeed, in [31] the assignment is autonomously performed by the vehicles that iteratively solve some Local Integer Linear Programming (L-ILP) problems, while our approach is based on heuristics that require a lower computational power.

In [32] authors deal with the control and collision avoidance of free-ranging AGVs removing the assumption on the existence predefined guide-path.

The work presented in [33] describes a decentralized control architecture for coordinating vehicles moving along a network of interconnected predefined path regions. With respect to such an approach, in this work vehicles are assumed to be autonomous and not controlled by distributed regional controllers and, consequently, there is no need for the installation of additional infrastructure into the working environment.

With respect to [34], where a decentralized coordination algorithm for safe and efficient traffic management of vehicles moving within a dynamic industrial environment is proposed, in this work a structured environment is assumed: vehicles move along a monodimensional guide-path autonomously regulating their speed for collision avoidance.

With respect to game-tehory methods of [39] and [40] that can be applied in the studied application domain, the proposed frame-

work makes it possible to quickly assign a mission and each vehicle does not have to constantly communicate with its neighbourhood to coordinate its movements.

In [28] an on-line supervisory control approach, based on a limited look-ahead policy for the control of multi-agent discrete-event systems (DESs) is shown. However, as discussed in this work, look-ahead policies can be computationally expensive and therefore not suitable for AWSs especially when near-real-time constraints must be taken into account. Indeed, in order to allocate a mission (i.e. to choose "Who" must execute it and in "Which Way") heuristics are used. This is also due to the fact that problems in logistic systems are often combinatorial [25] with the property that as the number of possible solutions (combinations) increases, the computational time required to enumerate all these solutions (and to find the "optimal" one) exponentially grows under a computational complexity point of view. Moreover, note that a very large number of logistic problems is NP-Hard [25]. Furthermore, the use of heuristics is also motivated by the following factors [41]:

1. all the decisions must be accomplished in real-time;
2. the interactions among the vehicles and their routes make difficult to find the optimal solution;
3. workers' interactions with the system may have unpredictable response times to some actions.

The problem of recharging vehicles is not addressed in this work but, recharging paheses can be planned through auctions or decentralised strategies, as the one in [42].

Chapter 2

Control system architecture

In this section, the typical hierarchical and centralized control architectures used in AWSs and the proposed auction-based one are described. In detail, in Sec. 2.1 the differences between the two control architectures are highlighted, underlining their positive aspects and disadvantages. In addition, in Sec. 2.2, the IS of a modern AWS and the behaviour of each vehicle which travels along the mono-dimensional guide-path of such an IS is also depicted.

2.1 Control Architectures

In typical hierarchical and centralized control architectures employed in AWSs, vehicles maintains a connection to a central control unit that, on the basis of the information they provide, allocates the missions while also sending the appropriate commands to them in order to execute the assigned tasks. In detail, such control architectures usually consist of three different parts placed at three different levels [26]:

- Level 3: Planning System (PS);
- Level 2: Management System (MS);

- Level 1: Handling System (HS).

In detail, the MS receives information from the PS about the load/unload operations. Then, on the basis of the location map (i.e. the list of the SUs currently stored in the AWS and their coordinates), it computes the list of missions to accomplish. Such a list is forwarded to the HS that, according to the global state of the system and to a predefined optimization policy, reschedules and segments them into basic handling sequences that are consequently executed. In [26] authors propose to separate the handling phase from the optimization phase by introducing another intermediate level, the Optimiser System (OS), between MS and HS: this reflects the recent trend to bridge the gap between the planning/management system and the control system of an AWS using on-line information to manage the current application of production resources. More specific:

- PS does not know where a SU is, but it only knows whether a SU is or is not in the warehouse, thus it can order to pick it by sending a retrieval mission to the MS.

- PS does not know where an SU has to be stored but it only knows that there is at least an empty location in the warehouse, thus it can send a storage mission to the MS.

- PS works on a statistical characterization of the system performance. Its main task is material planning and lot-sizing keeping in mind the constraints concerning the inventory levels, buffer capacities and production times.

- The MS is not aware of the actual state of the whole system and it schedules the missions to be performed exclusively on the basis of the location map by using its optimal management algorithm.

- The OS is dedicated to the control and coordination of the AWS devices (conveyors, buffers, cranes, vehicles, etc.) and

has to manage multiple missions at the same time. In addition, it is responsible to divide each mission in basic handling sequences (e.g. vehicle V_i goes to position p in the zone Z_j).

- The HS executes all the basic handling sequences provided by the OS.

Moreover, there is also a flow of data that starts from the HS and arrives to the PS to update the inventory of the warehouse.

In the just explained control architecture, a vehicle is not aware of its destination and it is not aware of the vehicles in its surroundings. Indeed, the allocation of a mission and the movements of vehicles are managed by the OS and the HS respectively: on the basis of the state of the whole system and of user-defined heuristics, the path the vehicles follow is dynamically computed [43]. Note that the central control unit (composed by the 3 levels described above) always interacts with each vehicle of the system managing the task allocation process and the path of each vehicle. Even if the advantage of these approaches is the reduction of efforts during the system design phases, resulting in cost and time savings [44], it is strictly constrained by the computational time requested for the real-time motion of the vehicles, that increases with the number of the vehicles involved [34].

In auction-based approaches, vehicles are autonomous entities aware of their state, their current destination and the zone they are traversing. With the respect to the centralized approach, each vehicle owns a modular implementation of the MS, OS and of the HS allowing:

- to perform the mission allocation process through the exchange of messages;

- each of them to autonomously decide the path to follow.

An high-level view of a traditional centralized control architecture of an AWS is reported in Fig. 2.1 together with its auction-based implementation. In such a figure, each of the N_v vehicles

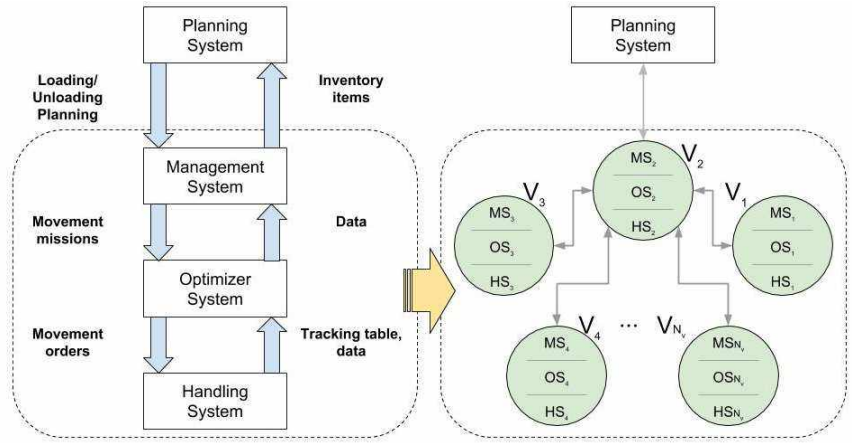


Figure 2.1: Typical three-level hierarchical and centralized control architecture of an AWS and its auction-based implementation where V_2 plays the role of the auctioneer.

is represented as a green circle and grey arrows represent the exchange of messages among them.

In auction-based approaches, an entity (i.e. a vehicle in this case) acts as an "auctioneer" which plays as a central coordinator. Different auction design methods have been proposed in the literature but, the most adopted high-level protocol to successfully accomplish an auction is the so called Contract Net Protocol (CNP): it specifies the interaction between entities for autonomous competitive negotiation through the use of contracts [45]. The CNP allows a flexible distribution of information through three methods:

- entities can transmit a request directly to another entity for the transfer of the required information;
- entities can broadcast a task announcement in which the task is a transfer of information;
- entities can observe, in its bid on a task, that they require particular information in order to execute the task.

The CNP works as follows:

- *Announcement stage*: the auctioneer forwards the mission to the other vehicles;
- *Submission stage*: all the vehicles (or a subset of them) reply to the auctioneer with a bid on the mission (that is, in most of cases, a function of the utility);
- *Selection stage*: the auctioneer evaluates all the received bids and selects the vehicle that must perform the mission optimizing a predefined global objective function;
- *Contract stage*: the auctioneer forwards a message in which it declares the winner.

Obviously, further rules and/or phases can be added to the above steps and different types of protocols can be exploited to exchange messages among the vehicles.

In the proposed auction-based control architecture, as it will be also discussed subsequently, only the strictly necessary amount of information is exchanged in each phase: during the announcement stage, the auctioneer forwards to each vehicle the id of the mission together with the picking and deposit locations of the corresponding SU; in the submission stage, each vehicle analyses such information and reply with a bid that depends on the used heuristics; in the selection stage, no messages are exchanged since the auctioneer decides the vehicle that will accomplish the mission; in the last stage the id of the chosen vehicle is sent to the entire fleet. Moreover, it is relevant to point out that each vehicle is considered as a self-interested entity: this means that the final solution may be optimal for the winner vehicle but not for the entire group. In Fig. 2.2 all the phases of the auction-based mechanism are represented. The announcement stage, the submission stage, the selection stage and the contract stage are represented in red, green, blue and purple respectively. In such a figure, every time a mission must be allocated, its details are forwarded to all the vehicles and, on the basis of such details and, again, on

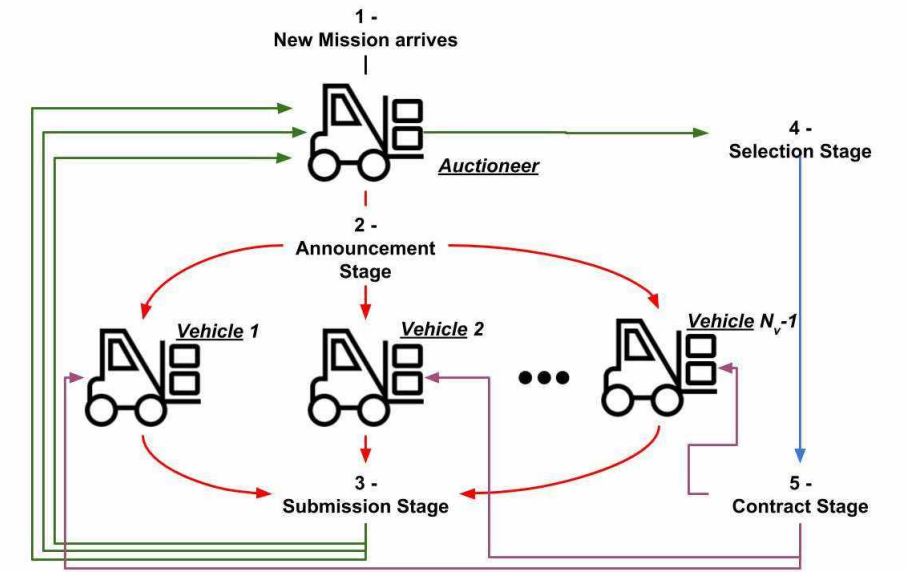


Figure 2.2: Example of the auction process composed by the announcement stage in which the auctioneer forwards all the details about a mission to all the vehicles of the system; the submission stage in which all the vehicles reply with their utility to the auctioneer; the selection stage in which the auctioneer selects the winner vehicle; the contract stage in which the auctioneer alerts the other vehicles about the winner.

user-defined heuristics, a vehicle is chosen and its controller takes decisions about the next zone to enter in (see. Sec. 2.2 for more details). In this work, utility and bid are used as a synonym.

2.2 Interface System and vehicles behaviour

The IS is (usually) composed by two interacting subsystems namely "vehicle subsystem" and "guide-path subsystem". The first is composed by vehicles which transport SUs from picking bays to deposit bays and viceversa, the second is characterized by the

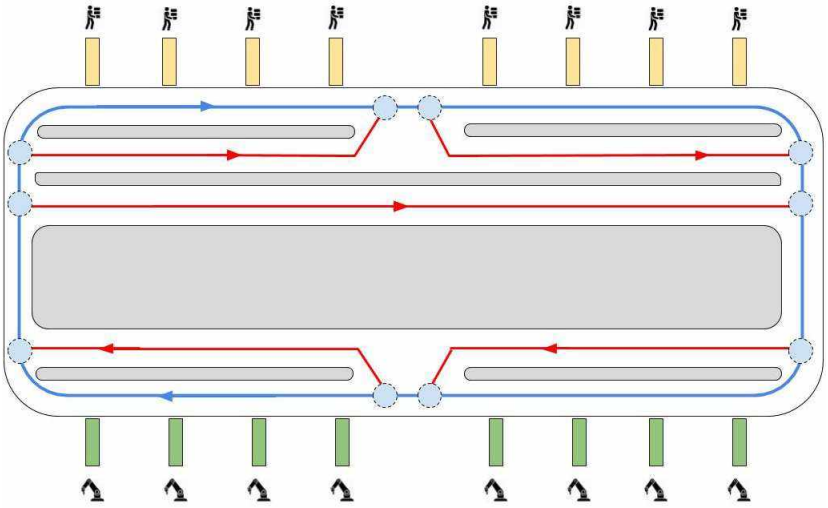


Figure 2.3: Example scheme of a real IS. Picking bays and deposit bays are shown as yellow and green rectangles respectively; the guide-path subsystem is shown with blue and red lines and the arrows represent the driving direction of the vehicles. Moreover, branch-points are depicted as blue circles.

mono-dimensional guide-path on which vehicles can move. Such guide-path is divided into zones (also called routes). The point where a zone ends and another zone starts is called branch-point. Exchange points represent points of the interest of the guide-path where vehicles can perform picking or deposit actions.

Branch-points and exchange points represent two critical aspects of the guide-path subsystem:

- the first allows a vehicle to exit from a zone end enter in one of the possible adjacent zones;
- the second allows a vehicle to change its destination (e.g. a vehicle is set busy and its destination becomes the bay where it has to unload its assigned SU).

It is worth mentioning that, because of the presence of branch-points, vehicles can choose among different paths to reach a certain destination point.

In Fig. 2.3 an example scheme of a real IS is shown. Picking bays and deposit bays are shown as yellow and green rectangles respectively. The guide-path subsystem is shown with blue and red lines. The arrows represent the driving direction of the vehicles. The red lines represent routes that can be used to reach the exchange points without passing near any bay. The blue lines represent an external circuit that vehicles can use to perform loading (unloading) operations. Branch-points are represented with blue circles.

Let N_v be the number of vehicles of the IS. Each vehicle is assumed to be equipped with a robust feedback control system so that the desired behavior can be assumed to be the actual behavior. Let $\boldsymbol{x} = [x_1 \ x_2]^T$ be the state vector of the r-th vehicle V_r of the system, where x_1 and x_2 represent its position and its velocity respectively. The desired behavior of the controlled system results to be [46]:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}a \tag{2.1}$$

where $\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $\boldsymbol{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and a is the acceleration.

The details of control system are outside the scope of this work, while in the following we describe in detail the controlled behaviour.

The controlled behaviour of V_r, that is represented in the net model, can be summarized as follows:

1. V_r can move with a uniformly accelerated motion until it reaches its maximum velocity v_{max};

2. V_r can move with constant speed motion when it reaches v_{max};

3. V_r must maintain a safety distance from other vehicles to avoid collisions;

4. V_r must stop at the exchange points to perform a load (unload) operation.

Moreover, V_r must regulate its velocity to avoid collisions: this means that the distance between two successive vehicles must always be greater than a predefined value Th (behaviour 3 of the vehicle subsystem). In addition, if it is requested to perform a load (unload) operation, V_r starts decelerating with constant deceleration up to the position pos_d to stop at a pre-selected exchange point (behaviour 4 of the vehicle subsystem). When such an exchange point is reached, V_r waits δ_{stop} unit of time to simulate the exchange operation and then it starts to move again.

Let N_z be the number of zones of the guide-path subsystem. The constraints related to each zone Z_i can be summarized as follows:

1. Z_i can be crossed by more vehicles at a time;
2. Z_i can be linked with more than one adjacent zone.

Chapter 3

Auction-based Control algorithms

In this section, the devised control algorithms are detailed. As discussed in the previous sections, they make use of heuristics in order to decide "Who" must perform a certain mission and in "Which Way". In Sec. 3.1, an overview of the proposed heuristics is presented: they are inspired by the most common ones used in AWSs as well as by the standard auction types discussed in Sec. 1. In addition, in Sec. 3.2 and in Sec. 3.3, the centralized control algorithm and the auction-based one are detailed respectively and in Sec. 3.4 computational and communication issues are discussed for both the employed approaches.

3.1 Simulations and Heuristics in the loop

In order to decide "Who" must perform a certain mission, the subsequent heuristics are taken into account:

- H_{who_1}: "Choose the closest vehicle from the picking bay of the mission";

- H_{who_2}: "Choose the closest vehicle that will reach the picking bay of the mission using the least crowded path";
- H_{who_3}: "Choose the vehicle associated to the maximum priority zone";
- H_{who_4}: "Choose the vehicle that completes the mission first using forward simulations".

The first heuristic can be applied in both the centralized and the auction-based control architecture since the topology (layout) of the IS is known by the central control unit in the first case as well as by each vehicle in the second case. Thus, when there is the need to allocate a mission, the shortest distance between the current position of a vehicle to the picking bay of the mission is computed and the closest vehicle from such a picking bay is consequently chosen.

The second heuristic can be used only in the centralized control architecture because it requires the information about the current position of each vehicle as well as the path each vehicle is travelling.

The third heuristic is used only in the auction-based approach. It is a simplified version of H_{who_1} and has been designed to be implemented on the low-power control device of each vehicle as they only need to communicate their position to the auctioneer. Basically, when a new mission arrives, the auctioneer assigns a high priority to the zones closest to the picking bay and a low priority to the more distant ones. During the submission stage, each vehicle sets its bid equal to its position and communicates this value to the auctioneer that will assign the mission to the vehicle in the highest priority zone.

Finally, the last heuristic is exclusively used in the centralized control architecture and relies on forward simulations: when a mission arrives, the time that each vehicle spends to complete it is simulated (also taking into account all the missions in progress) and the mission is consequently assigned to the vehicle with the minimum completion time. However, note that the number of

simulations can be very high: if k is the number of missions to accomplish and n is the number of vehicles, then the number of possible forward simulations is n^k. For this reason, only a subset of the whole set of possible simulations are taken into account.

To decide in "Which Way" a SU must be transported from one bay to another, the following heuristics are used:

- H_{way_1}: "Choose the shortest path to go from the picking bay to the deposit bay of the mission";
- H_{way_2}: "Choose the least crowded path to go from the picking bay to the deposit bay of the mission".

The first heuristic, as discussed for H_{who_1}, is applied in both the approaches since it is based on the evaluation of a distance between two points along the guide-path.

Concerning the second heuristic, as discussed for H_{who_2}, the less crowded path can be computed only by the central entity and it is not used in the proposed auction-based approach.

3.2 Centralized approach

In this section, the centralized control algorithm is detailed. A first-in-first-out queue M_{list}, is used to manage all the missions to perform by the PS. Moreover, let H_{who} and H_{way} be the user-defined heuristics which describe "Who" must perform a mission and in "Which Way" respectively. In addition, let V_{free} be the list used to keep track of free vehicles of the system.

Algorithm 1 is employed to allocate missions to vehicles and works as follows. When $H_{who} = H_{who_1}$, the central control unit selects the closest vehicle to the picking bay of the incoming mission $m \in M_{list}$; otherwise, if $H_{who} = H_{who_2}$, the vehicle that will travel along the least crowded path to reach the same picking bay is chosen. If $H_{who} = H_{who_4}$, then forward simulations are used to assign missions to vehicles. At this aim, since the total number of possible forward simulations, as previously mentioned, is exponential in the number of vehicles and incoming missions, in general it is

Input: M_{list}, H_{who}, H_{way}, N_{max_fw}, V_{free}.
Output: Allocation of the mission $m \in M_{list}$.
if $H_{who} \in \{H_{who_1}, H_{who_2}\}$ **then**
  execute Algorithm 2 using the first mission $m \in M_{list}$ as input;
else
  for *each vehicle* V_r **do**
    if *(V_r must perform an unload operation and must not perform any other mission) or (V_r is free)* **then**
      put V_r in the list CAN of candidate vehicles for which a forward simulation must be performed;
    end
  end
  compute the set S of all the possible simulations that can be performed using the vehicles in CAN and missions in M_{list};
  if $|S| \leq N_{max\_fw}$ **then**
    perform the forward simulations;
    allocate the missions according to the results of the forward simulations;
  else
    execute Algorithm 2 using H_{who_1} as input;
  end
end

Algorithm 1: Centralized control algorithm.

not possible to perform an exhaustive analysis, and so only a subset of the whole set of possible simulations is considered, precisely the parameter N_{max_fw} denotes the maximum number of forward simulations to be executed. To set N_{max_fw}, some considerations are in order:

- the set of simulation must be completed in a time negligible with respect to the mean duration of a mission;
- only vehicles that are free or are transporting a SU to the corresponding deposit bay are considered eligible since they can start a mission immediately or shortly.

However, when the number of forward simulations to perform is greater than N_{max_fw}, Algorithm 2 with $H_{who} = H_{who_1}$ is executed.

Moreover, to take into account that the system state changes during the simulation phase, the initial state of each forward simulation will be set to the state the system will reach at the end of the simulation phase. Indeed, such a state can be obtained (estimated) by simulation using the hybrid model proposed in the work and taking into account the mean duration of a simulation and the total number of simulation to perform.

When a vehicle employed in a mission withdraws its assigned SU, Algorithm 3 is used to choose the path form the vehicle's current position to the corresponding deposit bay. If H_{who_4} is employed, such an algorithm receives as input $H_{way} = H_{way_1}$.

Details about the data structures employed to implement all the algorithms are not reported. This is due to the fact that for example, the information about free vehicles or information about who must perform a deposit action can be stored in a lot of different ways (e.g. lists, maps). Moreover, note that all the algorithms are executed in a control-loop in which the evolution of both the discrete and the continuous components of the system is followed (see Sec. 4 for further details). Also note that while changes in the discrete part happen only when particular events occur, continuous state changes nonstop. To guarantee the convergence of the algorithms, functions are implemented in the defined control loop to evaluate 1) if there are missions to allocate, 2) if all the vehicles have performed all the assigned missions and 3) if a maximum simulation time has been reached.

3.3 Auction-based approach

Before presenting the proposed auction-based control algorithm, let's generalize the heuristic H_{who_1} in order to select the k-th closest vehicle at distance r_p from the picking bay of a mission $m \in M_{list}$. Such a generalization results very useful in order to discuss the main proprieties of the proposed algorithm and to show,

Input:H_{who}, V_{free}.
Output: $V_r \in V_{free}$ that must execute the mission m (if any).
if $|V_{free}| \neq 0$ **then**
  Use H_{who} to select the vehicle $V_r \in V_{free}$;
else
  No vehicle V_r can be selected;
end

Algorithm 2: Use of the heuristics H_{who} to allocate a mission m to a vehicle V_r.

Input:H_{way}.
Output:Path for the vehicle V_r to reach the deposit bay of the mission m.
Use H_{way} to select the path to reach the mission deposit bay;

Algorithm 3: Use of the heuristics H_{way} to find the path to the mission deposit bay.

in Sec.5.2, that a design methodology can be devised for these approaches in AWSs. To this aim, the heuristic H_{who_1} is revised as follows:

- $H_{who_{1b}}$: "Choose the k-th closest vehicle at distance r_p from the picking bay of the mission".

Notice that, denoting with L_{max} the maximum distance a vehicle can be from another one, when $k = 1$ and $r_p = L_{max}$, the heuristic $H_{who_{1b}}$ reflects H_{who_1}.

Algorithm 4 details all the steps used to perform an auction to allocate missions to vehicles. Let τ_s and τ_r be the time that the auctioneer takes to forward its messages to all the vehicles and the time that all the vehicles take to reply to the auctioneer respectively. In addition, let $m_1, \ldots, m_{b_s} \subseteq M_{list}$ a subset of missions that must be allocated. The algorithm works as follows: when a batch of b_s missions must be performed, the PS forwards all its details to a random auctioneer. At this point all the stages of the auction-based mechanism start: the auctioneer sends a message to all the vehicles and, depending from the adopted heuristic,

Data: $\{m_1, \ldots, m_{b_s}\}$, H_{who}, r_p.
Result: allocation of $\{m_1, \ldots, m_{b_s}\}$.
Announcement stage (in time τ_s): the auctioneer V_a forwards a message specifying the batch of missions $\{m_1, \ldots, m_{b_s}\}$ to all the vehicles;
Submission stage (in time τ_r): each vehicle V_i reads the message; and compute its utility u_i according to H_{who};
if $H_{who} = H_{who_{1b}}$ **then**
  if $u_i \leq r_p$ *and* V_i *is free* **then**
    V_i sends u_i to V_a;
  else
    V_i discards the message;
  end
else
  V_i sends u_i to V_a
end
if *at least one vehicle has replied* **then**
  Selection stage: V_a applies H_{who} to decide the winner;
  Contract stage (in time τ_s): V_a announces the winner;
end

Algorithm 4: General k-price sealed-bid auction-based control algorithm with reserve price.

a group of them or the entire fleet reply to the auctioneer with its utility. In detail, if $H_{who} = H_{who_1}$, the bid of each vehicle is represented by the shortest distance between its current position and the picking bay of the mission; if $H_{who} = H_{who_3}$, each vehicle needs to communicate only its position without performing any other computation. The auctioneer locally collects information about which vehicle can perform the incoming mission and on the basis of all the values received, it declares the winner. Specifically, if $H_{who} = H_{who_1}$, the auctioneer selects the k-th vehicle to which the lowest bid is associated; if $H_{who} = H_{who_3}$, it selects the vehicle in the highest priority zone. Notice that when a vehicle has loaded its assigned SU, automatically perform Algorithm 3 with $H_{way} = H_{way_1}$ to choose the best path to follow to reach its destination. In addition, observe that the PS can be updated during

the contract stage: it is responsibility of the auctioneer to notify the PS if a mission is successfully allocated.

Algorithm 4 captures most of the concepts behind the k-price sealed-bid auctions while introducing some modifications and improvements inspired by other auction variants in the literature. Indeed:

1. k permits to establish which vehicle performs the missions and it is aimed at choosing a vehicle which, at the end of an auction, does not exceed the exchange point where a SU is to be withdrawn;

2. r_p acts as a "reserve price" for vehicles and permits to consider only a subset of them to communicate with;

3. b_s allows the use of multi-item auctions.

In addition to k, r_p and b_s, the number of the vehicles N_v is the fourth design parameter of the algorithm, while the response time τ, that is the response time of the allocation process, is a design constraint. In the case study, it will be shown that this set of 4 parameters can be tuned by means of k, r_p/L_{max}, b_s and N_v/N_{max} where N_{max} is the maximum number of vehicles the system can have with respect to the length of the circuit. This set of parameters is fully dimensionless and this allows to make general (and not ad hoc) the approach proposed in this work.

Regarding the response time τ of Algorithm 4, it is

$$\tau = 2(N_v - 1)\tau_s + N_v\tau_r + N_v\tau_{bid} + \tau_{win}$$

where τ_{bid} is the time spent by each vehicle to compute a bid and τ_{win} is the time spent by the auctioneer to calculate the winner of an auction. Notice that τ_{win} can be realistically assumed negligible respect to τ_{bid}. Thus,

$$\tau \approx 2(N_v - 1)\tau_s + N_v\tau_r + N_v\tau_{bid}$$

results to be dependant on τ_s and τ_r and on τ_{bid}, that is related to the on-board computation capability of the vehicles and the

computation effort of the algorithm for which a polynomial complexity can be obtained. Notice that when $\tau_s = \tau_r$, the response time becomes:

$$\tau \approx (3N_v - 2)\tau_s + N_v\tau_{bid}$$

3.4 Computational and communication issues

3.4.0.1 Computational issues

To establish "Who" has to perform a given mission and in "Which Way", in the auction-based control architecture each vehicle must be able to:

- compute the most efficient path to reach the picking bay of the mission from its current position;
- compute the most efficient path to reach the deposit bay of the mission from the corresponding picking bay;
- evaluate which is the most suitable vehicle for the mission if it is the auctioneer.

Because of the presence of branch points, there can be many possible ways for a vehicle to go from a certain origin position to a certain destination. In the auction-based approach it is supposed that vehicles are aware of the topology of the IS. One of the most effective ways to represent the topology of a map is using graphs. Many computationally efficient algorithms have been developed to work with this data structure. In this work, the well known Breadth-first search (BFS) algorithm is adopted for searching paths among the IS. In detail, each time there is the need to estimate the cost to go from a point to another, the BFS algorithm is used to compute each possible route to reach the destination and then, on the found routes, a predefined heuristics is applied. In the most general case, when an adjacency list is used as data structure to represent the graph, the time complexity of

this algorithm is $\mathcal{O}(|V|+|E|)$ where $|V|$ is the number of nodes (i.e. branch-points) and $|E|$ is the number of edges (i.e. routes which connect two consecutive branch-points). Note that the BFS algorithm is used both in the auction-based approach and in the centralized one.

To evaluate which is the most suitable vehicle to perform a mission, in the proposed control architecture the auctioneer must be able to compute the minimum/maximum among (at most) $N_v - 1$ received bids. Thus, it is possible to state that: 1) the announcement stage of the auction-based approach has a computational complexity of $\mathcal{O}(N_v)$; 2) the submission stage takes $\mathcal{O}(N_v \times (|V|+|E|))$ time; 3) the selection stage takes $\mathcal{O}(N_v)$ time; 4) the contract stage is performed in $\mathcal{O}(N_v)$ time. Consequently, the computational complexity of Algorithm 4 is $\mathcal{O}(N_v \times (|V|+|E|))$.

The computational complexity of Algorithm 1 strongly depends from the employed heuristics: when H_{who_1} and H_{who_2} are used, the computational complexity results in $\mathcal{O}(N_v \times (|V|+|E|))$; when H_{who_4} is employed, an exponential execution time is obtained since there is the need to enumerate all the possible forward simulations that must be accomplished and execute them when possible.

Algorithm 3, employed by both approaches to choose the path for a vehicle to reach its destination, exposes a time complexity of $\mathcal{O}(|V|+|E|)$.

3.4.0.2 Communication issues

In the auction-based approach, each vehicle of the IS must be able to communicate with the auctioneer in order to express its interest in a mission. When a mission arrives, the auctioneer forwards a number of messages equals to $N_v - 1$ to propagate the information about the mission. The forwarded information in each message are: the id of the mission (integer), the picking zone (integer) and the picking bay position (float) of the SU that must be transported and the deposit zone (integer) and the deposit position (float) of the same SU. So, five numbers (3 integers and 2 floats) must be ex-

changed during the announcement stage. In the submission stage, each vehicle will answer with its bid (float) that is computed taking in consideration the adopted heuristics. During the selection stage no messages are exchanged. Finally, in the last stage, the auctioneer sends $N_v - 1$ messages in which it forwards the id (integer) of the winner vehicle. For a performance optimization it can be stated that if a vehicle does not receive a message in a certain time range, it can suppose that the mission is not allocated to it. Without optimization a total of $(N_v - 1) \times 3$ messages must be exchanged for a successful allocation ($(N_v - 1) \times 2$ otherwise). However, it is licit to suppose that such amount of messages is not a problem: in AWSs the number of vehicles is limited and communication is not considered a bottleneck of such systems.

In the centralized approaches communication issues are less constrained. Indeed, the central control unit is always aware about the state of each vehicle and there is not necessity to exchange messages in order to perform the task allocation. Without loss of generality, there is the implicit assumption that a task allocation process is instantaneous in centralized approaches.

Chapter 4

A hybrid model to support the development of auction-based control algorithms

In this section, the hybrid model used to support the development of auction-based control algorithms is presented. To model the behaviour of the IS, Colored Modified Hybrid Petri Nets (CMHPNs) are employed and the motiviation behind their adoption are reported in Sec. 4.1. The whole IS model is composed by two parts: the first represents the vehicle's subsystem, the second the guide-path subsystem. In order to describe them, firstly, in Sec. 4.2, a background on CMHPNs is provided. Moreover, in Sec. 4.3.1 and in Sec. 4.3.2 details about the two subsystems are reported respectively. In addition to [4], the formal model presented in this section is conceived to be used together with an automated model generation algorithm. This is essential to support by means of simulations the design of control algorithms based on heuristics to be implemented in an auction-based control architecture. Thus, in Sec. 4.4, details about the algorithm that allows the automatic generation of the model of the whole IS are provided.

4.1 Motivation behind the adoption of CMHPNs

The analysis and control of AWSs are very complex: the set of constraints that characterize such systems (e.g. availability of resources, interaction between subsystems, variation of the number of the resources) makes crucial the choice of the model to be used in each step of control system development [25]. Due to the complexity of the interrelationships, an effective analysis of the impact of different control strategies on the system must consider each component and smart device not in isolation from each other but rather as an integrated whole (see [47] in the context of a baggage handling system).

Depending on the dynamics of the system, different types of models can be adopted in the logistic domains [25]. Examples are: Markovian processes, PNs (with their variants) and automata. PNs have been successfully used to model and analyze many DESs (e.g. automated storage and retrieval system of [30], large scale automated manufacturing systems of [48], a treelike hybrid multicluster tool to find its optimal one-wafer cyclic schedule [49], UML modelled systems [50], traffic sysytems [51]) thanks to their powerful mathematical formalism that enables both qualitative and quantitative analysis, and also the synthesis of supervisors to ensure deadlock recovery [52] or liveness [53].

In this work CMHPNs are used to model the IS of an AWS as done in [4] to:

- model picking and deposit actions of each vehicle and the switching between different vehicle's dynamics as a DES;
- model the dynamic of each vehicle as a continuous system.

The developed model is highly modular and compact. The first characteristic is due to the identification of elementary modules of the systems that can be concatenated to obtain different typologies of ISs in an easy manner. The overall system model is obtained by composing elementary modules according to the constraints

represented by vehicles and routes. Compactness is achieved by introducing colors and structured markings in the Hybrid Petri net (HPN) model. Moreover, the use of a PN model, permits to provide local state representation [30, 54] and allows the model of the whole IS to be automatically synthesized. Furthermore, it is also possible to formally characterize the system state, to use standard simulation schemes and to realize shut-down and recovery mechanisms so that the system state and all simulation outputs are saved at a given time and restored when necessary.

4.2 Background on CMHPNs

A CMHPN model merges the concepts of Modified Hybrid Petri Nets (MHPNs) and Colored Petri Nets (CPNs).

Formally, a HPN is a 7-tuple $\{P, T, \boldsymbol{Pre}, \boldsymbol{Post}, h, \boldsymbol{\delta}, \boldsymbol{\nu}\}$ where:

- $P = P^D \cup P^C$ is the set of discrete places P^D and continuous places P^C such that $P^D \cap P^C = \emptyset$ and $|P^D| = w_d$, $|P^C| = w_c$, $|P| = w = w_d + w_c$;
- $T = T^D \cup T^C$ is the set of discrete transitions T^D and continuous transitions T^C such that $T^D \cap T^C = \emptyset$ and $|T^D| = n_d$, $|T^C| = n_c$, $|T| = n = n_d + n_c$;
- $\boldsymbol{Pre} : P \times T \to \mathbb{R}^+$ is the $(w \times n)$-size pre-incidence matrix;
- $\boldsymbol{Post} : P \times T \to \mathbb{R}^+$ is the $(w \times n)$-size post-incidence matrix;
- $h : P \cup T \to \{D, C\}$ is the "hybrid function" which indicates, for each node, if it is of a discrete or continuous type;
- $\boldsymbol{\delta} : T^D \to (\mathbb{R}^+)^{n_d}$ is the n_d-size firing delay vector where:

$$\delta_j = \begin{cases} 0 & \text{if } t_j^D \text{ is immediate} \\ > 0 & \text{if } t_j^D \text{ is timed} \end{cases}$$

  is the firing delay related to the each discrete transition t_j^D (with $1 \leq j \leq n_d$);

- $\boldsymbol{\nu} : T^C \to (\mathbb{R}^+)^{n_c}$ is the n_c-size firing speed vector where:

$$\nu_j = \begin{cases} 0 & \text{if } t_j^C \text{ is disabled} \\ \bar{\nu} & \text{if } t_j^C \text{ is enabled} \end{cases}$$

  is the firing speed associated to the continuous transition t_j^C (with $1 \leq j \leq n_c$) and $\bar{\nu}$ is the maximal firing speed associated to the same transition.

  For the sake of clarity, in this work, discrete and continuous places will be graphically represented with one and two line circles respectively; discrete and continues transitions will be drawn with one and two line rectangles respectively. Moreover, discrete immediate transitions are graphically represented with black bars and, discrete timed transitions are represented with white bars.

The $(w \times n)$-size incidence matrix of the net $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ can be written as

$$\boldsymbol{C} = \begin{pmatrix} \boldsymbol{C}_{CC} & \boldsymbol{C}_{CD} \\ \boldsymbol{C}_{DC} & \boldsymbol{C}_{DD} \end{pmatrix} \tag{4.1}$$

where the first block $\boldsymbol{C}_{CC}$ regards all the connections between continuous nodes; the second block $\boldsymbol{C}_{CD}$ regards all the connections between continuous places and discrete transitions; the third block $\boldsymbol{C}_{DC}$ regards all the connections between discrete places and continuous transitions and the last block refers to all the connections between discrete nodes. Notice that the firing of continuous transitions cannot change the marking of discrete places. Thus, $\boldsymbol{C}_{DC} = \boldsymbol{0}$.

A transition $t_j^D \in T^D$ can be autonomous or non-autonomous. In the first case, t_j^D is not associated to any logical expression $b : T^D \to \{0, 1\}$; in the second case it is. Such a logical expression is defined as a boolean function that can depend on other two boolean functions: one related to an external control input g and one related to an internal condition e. When $g = 1$, it means that an external controller sets to true the exogenous event associated to the transition t_j^D; when $e = 1$, it means that the endogenous event associated to t_j^D is verified.

The state of the net at time τ_k is represented by the marking

$$\boldsymbol{m}(\tau_k) = \{\boldsymbol{m}^C(\tau_k), \boldsymbol{m}^D(\tau_k)\}$$

where $\boldsymbol{m}^C(\tau_k) : P^C \to \mathbb{R}^+$ is the marking of the continuous time subsystem at time τ_k and $\boldsymbol{m}^D(\tau_k) : P^D \to \mathbb{N}$ is the marking of the discrete events subsystem at the same time. Notice that in the first case the marking of each continuous place is characterized by a real number and, in the second case, the marking of each discrete place is characterized by a non negative integer number. The notations ${}^\bullet p({}^\bullet t)$ and $p^\bullet(t^\bullet)$ represent the preset and the postset of each place $p \in P$ (transition $t \in T$).

A discrete transition $t_j^D \in T^D$ is enabled at time τ_k if and only if:

$$m_p(\tau_k) \geq Pre(p, t_j^D), \forall p \in^\bullet t_j^D \tag{4.2}$$

where $m_p(\tau_k)$ represents the marking of each place $p \in^\bullet t_j^D$ at time τ_k. Moreover, a discrete transition $t_j^D \in T^D$ can fire if and only if Eq. 4.2 holds and the associated logical expression is true. A continuous transition t_j^C is enabled if and only if the following two condition hold:

$$\begin{aligned} &m^D_{p^D}(\tau_k) \geq Pre(p^D, t_j^C), && \forall p^D \in^\bullet t_j^C \\ &m^C_{p^C}(\tau_k) > 0, && \forall p^C \in^\bullet t_j^C \end{aligned}$$

The feeding speed of a continuous place $p^C \in P^C$ is defined as:

$$I = \sum_{t_j^C \in {}^\bullet p^C} Post(p^C, t_j^C)\nu_j$$

When $I > 0$, the continuous place p^C is said to be fed; the draining speed of a continuous place $p^C \in P^C$ is defined as:

$$O = \sum_{t_l^C \in p^{C\bullet}} Pre(p^C, t_l^C)\nu_l$$

The time derivative of the marking of a continuous place $p^C \in P^C$ is called balance and it is defined as:

$$\dot{\boldsymbol{m}}_{p^C} = I - O = \frac{d\boldsymbol{m}_{p^C}}{dt} \tag{4.3}$$

The fundamental equation which describes the evolution of the HPN can be written as:

$$\begin{bmatrix} \boldsymbol{m}^C(\tau_k) \\ \boldsymbol{m}^D(\tau_k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{m}^C(\tau_k - 1) \\ \boldsymbol{m}^D(\tau_k - 1) \end{bmatrix} + \begin{bmatrix} \boldsymbol{C}_{CC} & \boldsymbol{C}_{CD} \\ \mathbf{0} & \boldsymbol{C}_{DD} \end{bmatrix} \cdot \left(\begin{bmatrix} \mathbf{0} \\ \boldsymbol{\sigma}^D(\tau_k) - \boldsymbol{\sigma}^D(\tau_k - 1) \end{bmatrix} + \int_{\tau_{k-1}}^{\tau_k} \begin{bmatrix} \boldsymbol{\nu} \\ \mathbf{0} \end{bmatrix} \right) \tag{4.4}$$

where $\boldsymbol{\sigma}^D(\tau_k) : T^D \to \mathbb{N}^{n_d}$ is the n_d-size discrete firing vector and the j-th element $\sigma_j^D(\tau_k) \in \boldsymbol{\sigma}^D(\tau_k)$ represents the number of times that the discrete transition $t_j^D \in T^D$ has fired until the time τ_k.

A Modified Hybrid Petri Net (MHPN) is a HPN in which the maximal firing speed $\bar{\nu}_j$ associated to each continuous transition t_j^C is a function of 1) the input places marking, 2) the input vector and 3) the time (rather than a constant value). Thus:

$$\bar{\nu}_j(\tau) = f(\boldsymbol{m}(\tau), \boldsymbol{u}(\tau), \tau)$$

In this work, the function f is a linear function of $\boldsymbol{m}(\tau)$ and $\boldsymbol{u}(\tau)$ used to model the switching between different linear, time-invariant, continuous dynamics. To compact the state representation, colors and structured markings are adopted. The just defined network with the above described conventions is called CMHPN. In the following, the structured marking of a continuous place with respect to the color r will be denoted as $< attr_1, \ldots, attr_q >_r$ where $attr_1, \ldots, attr_q$ are the q attributes of such a structured marking. Moreover $(attr_k)_r$ will denote the k-th attribute of the structured marking with respect to the color r.

Formally a CMHPN is a 4-tuple $\{\mathcal{N}, Cl, Co, \boldsymbol{\nu}\}$, where:

- $\mathcal{N}$ is a HPN;
- Cl is the set of colors;
- $Co : P \cup T \to Cl$ is the "color function" that assigns to each place $p \in P$ and each transition $t \in T$ a color $c \in Cl$;

- $\boldsymbol{\nu} : Co(t_j^C) \rightarrow \mathbb{R}^+$ is the mapping that assigns to each color of the continuous transition $t_j^C \in T$ a firing speed.

The set of possible colors of $p_i \in P$ ($t_j \in T$) is $Co(p_i)$ ($Co(t_j)$) and $|Co(p_i)| = u_i \leq |Cl|$ ($|Co(t_j)| = v_j \leq |Cl|$).

The marking $\boldsymbol{m}_{p_i}^D : Co(p_i^D) \rightarrow \mathbb{N}$ is the mapping that associates, to each color of the discrete place p_i^D, a non negative integer number which represents the number of tokens of that color in such a place. The marking of a discrete place p_i^D with respect to the color r will be denoted in the following with c_r.

Logical expressions associated to each discrete transition t_j^D are now represented as v_j-size column vector where the r-th element is the logical expression associated to the same transition with respect to the color r.

The structured marking $\boldsymbol{m}_{p_i}^C : Co(p_i) \rightarrow (\mathbb{R}^+)^q$ is the mapping that associates to each continuous place p_i^C with respect to the color r a q-size attribute vector of non negative real numbers:

$$< attr_1, \ldots, attr_q >_r, \ \forall attr_k \in \mathbb{R}^+, 1 \leq k \leq q$$

The v_c-size vector ($v_c \leq v_j$) of firing speed $\boldsymbol{\nu}(t_j^C) = \boldsymbol{\nu}_j$ of the continuous transition t_j^C is $\boldsymbol{\nu}_j = (\nu_{j,1}, \ldots, \nu_{j,v_c})^T$ where $\nu_{j,r}$ is the firing speed of t_j^C with respect to the color r. Notice that when t_j^C is enabled under the color r at time τ_k, $\nu_{i,r} = f(\boldsymbol{m}(\tau_k), \boldsymbol{u}(\tau_k))$.

The v_d-size vector ($v_d \leq v_j$) of firing delays $\boldsymbol{\delta_j}(t_j^D) = \boldsymbol{\delta}_j$ of the discrete transition t_j^D is $\boldsymbol{\delta}_j = (\delta_{j,1}, \ldots, \delta_{j,v_d})^T$ where $\delta_{j,r}$ is the firing delay of t_j^D with respect to the color r.

The pre-incidence matrix is $\boldsymbol{Pre}(p_i, t_j) : Co(t_j) \rightarrow \mathbb{R}^+(Co(p_i))$ and the post-incidence matrix $\boldsymbol{Post}(p_i, t_j) : Co(t_j) \rightarrow \mathbb{R}^+(Co(p_i))$ $\forall i \in \{1, \ldots, w\}, j \in \{1, \ldots, n\}$. In addition, notice that $\boldsymbol{Pre}(p_i, t_j)$ ($\boldsymbol{Post}(p_i, t_j)$) is a $u_i \times v_j$ matrix and the value $\boldsymbol{Pre}(p_i, t_j)(r, s)$ ($\boldsymbol{Post}(p_i, t_j)(r, s)$) represents the weight of the arc that connects the place p_i (transition t_j) with respect to the color r (s) to the transition t_j (place p_j) with respect to the color s (r). The incidence matrix $\boldsymbol{C}$ of CMHPNs has the same form of the incidence matrix of Eq 4.1.

Event	Expression of the j-th element	Meaning
e_1	$< x_1, x_2 >_r \geq < Th, 0 > + < x_1, x_2 >_k$	The safety distance between the vehicle V_r and the vehicle V_k is maintained.
e_2	$< x_1, x_2 >_r = < x_1, v_{max} >$	Vehicle V_r has reached its maximum velocity.
e_3	$< x_1, x_2 >_r \leq < Th, 0 > + < x_1, x_2 >_k$	The safety distance between the vehicle V_r and the vehicle V_k is not maintained.
e_4	$< x_1, x_2 >_r = < x_1, 0 >$	Vehicle V_r has its velocity equal to 0.
e_5	$< x_1, x_2 >_r = < pos_d, x_2 >_r$	Vehicle V_r has reached pos_d.

Table 4.1: Internal conditions and their meaning

4.3 CMHPN model of the whole IS

4.3.1 Vehicle subsystem

As described in Sec. 2.2, the IS is composed by two interacting subsystems, namely "vehicle subsystem" and "guide-path subsystem". Thus, in order to derive the CMHPN model IS_{net} of the whole IS, the CMHPN model IS_v of the vehicle subsystem and the CMHPN model IS_g of the guide-path subsystem are firstly created and then appropriately concatenated.

The CMHPN IS_v of Fig. 4.1 models the vehicle subsystem. Colours are used to model more than one vehicle at a time: each node of such a net has as many colors as the number of the vehicles of the system. The structured marking $\boldsymbol{x} = < x_1, x_2 >$ of the colored continuous place p_s allows representing the state of each vehicle. In particular, the structured marking $\boldsymbol{x}_r = < x_1, x_2 >_r$ represents the state of the r-th vehicle V_r of the system. As it will be detailed in the Sec.4.3.2, $0 \leq (x_1)_r \leq L_i$, where L_i is the length of the zone Z_i in which the vehicle V_r is located. The colored continuous transitions t_{acc}, t_{const} and $t_{dec}(t_{exch})$ allows updating the state of each vehicle when it accelerates, proceeds at constant speed, and decelerates respectively. To maintain the model as clear as possible, all the events associated to each transition are reported in Tab.4.1 where the following notation is used: given two vectors $< x, y >$ and $< w, z >$, $< x, y > \pm < w, z > = < x \pm w, y \pm z >$ with $x, y, z, w \in \mathbb{R}$. Notice that the firing speed of the transitions

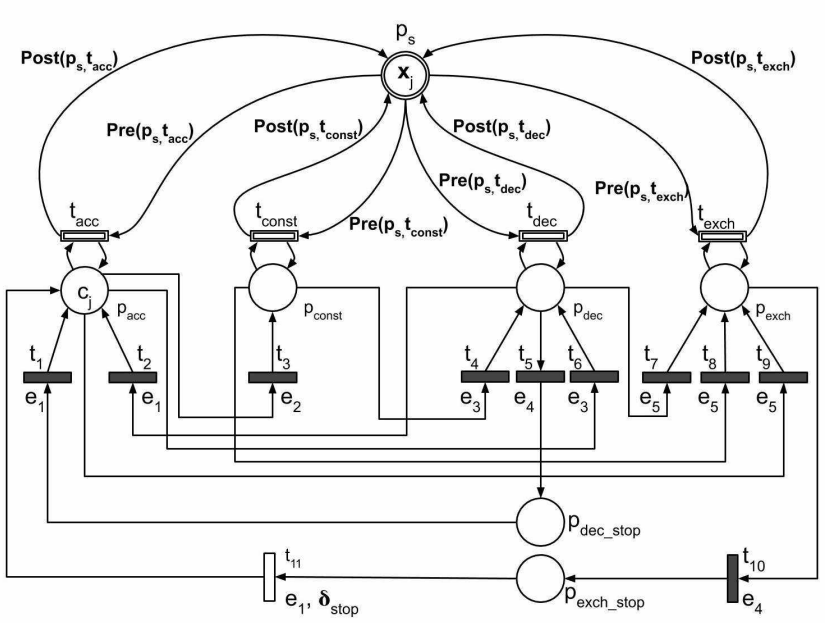


Figure 4.1: CMHPN IS_v used to model vehicles in the IS.

cited above is $\boldsymbol{A} \cdot \boldsymbol{x} + \boldsymbol{B} \cdot a$ or $\boldsymbol{A} \cdot \boldsymbol{x}$. Furthermore, notice that in order to correctly update the attributes of the colored continuous place p_s, it is necessary to add weights to the arcs connecting such a colored continuous place with each colored continuous transition. In detail, recalling Eq. 4.3, the following block diagonal matrices of weights are introduced:

$$\begin{aligned}
\mathbf{Post}(p_s, t_{acc}) &= I_{N_v} \otimes \boldsymbol{\pi_3} \\
\mathbf{Pre}(p_s, t_{acc}) &= I_{N_v} \otimes \boldsymbol{\pi_2} \\
\mathbf{Post}(p_s, t_{const}) &= I_{N_v} \otimes I_2 \\
\mathbf{Pre}(p_s, t_{const}) &= I_{N_v} \otimes \boldsymbol{\pi_2} \\
\mathbf{Post}(p_s, t_{dec}) &= I_{N_v} \otimes \boldsymbol{\pi_1} \\
\mathbf{Pre}(p_s, t_{dec}) &= I_{N_v} \otimes \boldsymbol{\pi_2} \\
\mathbf{Post}(p_s, t_{exch}) &= I_{N_v} \otimes \boldsymbol{\pi_1} \\
\mathbf{Pre}(p_s, t_{exch}) &= I_{N_v} \otimes \boldsymbol{\pi_2}
\end{aligned} \tag{4.5}$$

where the generic term I_p denotes the $p \times p$ identity matrix, $\otimes$

Name	Node	Type	Meaning
p_s	Place	Continuous	Models the state of a vehicle.
t_{acc}	Transition	Continuous	Vehicle is moving with uniformly accelerated motion.
t_{const}	Transition	Continuous	Vehicle is moving with constant speed.
t_{dec}	Transition	Continuous	Vehicle is moving with uniformly decelerated motion to avoid a collision.
t_{exch}	Transition	Continuous	Vehicle is moving with uniformly decelerated motion to stop at a certain exchange point.
p_{acc}	Place	Discrete	Enables uniformly accelerated motion.
p_{const}	Place	Discrete	Enables constant speed motion.
p_{dec}	Place	Discrete	Enables uniformly decelerated motion to avoid a collision.
p_{exch}	Place	Discrete	Enables uniformly decelerated motion to stop at a certain exchange point.
p_{dec_stop}	Place	Discrete	Models the stop of the vehicle to avoid a collision.
p_{exch_stop}	Place	Discrete	Model the stop of the vehicle to load/unload a SU.
t_1, t_2	Transition	Discrete	Enables the vehicle to move with a uniformly accelerated motion.
t_3	Transition	Discrete	Enables the vehicle to move with a constant speed motion.
t_4, t_5, t_6	Transition	Discrete	Enables the vehicle to move with a uniformly decelerated motion to avoid a collision.
t_7, t_8, t_9	Transition	Discrete	Enables the vehicle to move with a uniformly decelerated motion to stop at a certain exchange point.
t_{10}	Transition	Discrete	Enables the vehicle to perform a load/unload operation.
t_{11}	Transition	Discrete (Timed)	Enables the vehicle to move with a uniformly accelerated motion after performed a load/unload operation.

Table 4.2: Meaning of each place and transition of the proposed IS_v model of Fig. 4.1.

denotes the Kronecker product and:

$$\boldsymbol{\pi_1} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{\pi_2} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \boldsymbol{\pi_3} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}.$$

Consider the r-th vehicle of the system V_r. The colored continuous transition t_{acc} of color r can fire only when the marking of the colored discrete place p_{acc} is c_r. Such a place is marked when V_r moves according to Eq. 2.1. When p_{acc} is marked three events can occur: 1) V_r reaches its maximum velocity v_{max} (i.e. e_2 occurs); 2) V_r arrives in pos_d and must decelerate to stop at an exchange point (i.e. e_5 occurs); 3) the distance between V_r and its successor gets lower than the predefined threshold Th (i.e. e_3 occurs).

When e_2 occurs, the transition t_3 fires and the colored discrete place p_{const} becomes marked allowing the vehicle to move with the corresponding dynamics. When e_5 occurs, the transition t_9 fires and the place p_{exch} becomes marked. In such a case, V_r starts to decelerate in order to stop at the chosen exchange point to perform the pre-planned load (unload) operation. Finally, when the event e_3 occurs, the transition t_6 fires and the place p_{dec} becomes marked. Even in this case, V_r starts to decelerate to the aim of avoiding a collision. Notice that it can decelerate to a stop (i.e. p_{dec_stop} becomes marked) or, during its deceleration, can reach the position pos_d (i.e. e_5 occurs, the transition t_7 fires and p_{exch} becomes marked) and continue its deceleration until it will stop at the chosen exchange point. From p_{const} two events can occur: e_3 or e_5. The occurrence of the first or the second event enables the firing of transitions t_4 and t_8 respectively. When p_{exch} is marked (i.e. the vehicle is decelerating to reach the pre-assigned exchange point), the colored discrete transition t_{10} fires only when the speed of the vehicle is zero (i.e. the event e_4 occurs). Then, the vehicle waits δ_{stop} units of time (to simulate the load (unload) operation) and starts moving again. For seek of clarity, to better summarize the behaviour of each vehicle of the system, the meaning of each place and transition of the proposed IS_v model, is reported in Tab. 4.2.

4.3.2 Guide-path modeling

A guide-path is composed by one or more zones connected in a manner to form a circuit.

A generic zone Z_i with length L_i is modelled with a colored discrete place p_i. As in the vehicle subsystem, the number of colors of the CMHPN IS_g of the guide-path subsystem is equal to the number of vehicles of the system. The marking of p_i with respect to the color r indicates that the vehicle V_r is in the zone Z_i. In Fig. 4.2 an example of a CMHPN used to model a generic four-zones guide-path subsystem is represented. To connect two adjacent zones a colored discrete transition is used. For example,

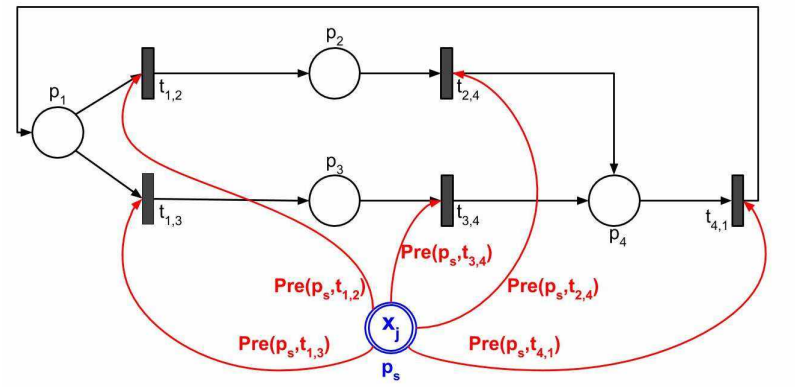


Figure 4.2: Example of a CMHPN IS_g used to model four zones of a generic IS. In red are shown the arcs used to link the such a net to the place p_s of the vehicle subsystem, drawn in blue.

always in Fig. 4.2, the colored discrete transition $t_{1,2}$ represents that the first zone and the second one are adjacent. In more general case a colored discrete transition $t_{i,k}$ is created to connect a zone Z_i with another adjacent zone Z_k. Thus, an arc from the colored discrete place p_i to the colored discrete transition $t_{i,k}$ and an arc from $t_{i,k}$ to the colored discrete place p_k is created. Furthermore, an arc from the colored continuous place p_s (of the vehicle subsystem) to each colored discrete transition $t_{i,k}$ must be added and weighted with the following block diagonal matrix:

$$\mathbf{Pre}(p_s, t_{i,k}) = \mathrm{diag}(L_i) \in \mathbb{R}^{N_v \times N_v} \tag{4.6}$$

This permits to:

- represent the position x_1 of each vehicle V_r in the generic zone Z_i with a value $0 \leq (x_1)_r \leq L_i$;
- connect the CMHPN model IS_v of the vehicle subsystem with the CMHPN model IS_g of the guide-path subsystem.

The second step is fundamental: it makes possible to obtain the model IS_{net} of the whole IS. Moreover, also notice that, in the example of Fig. 4.2, the colored discrete places p_1, p_2, p_3 and p_4 model

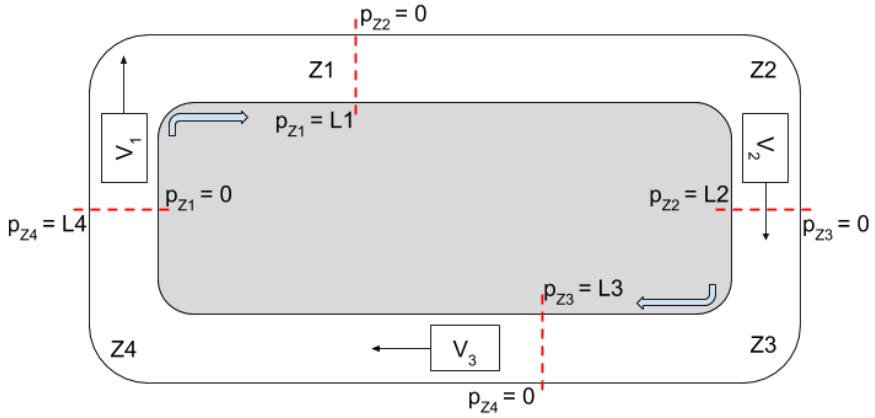
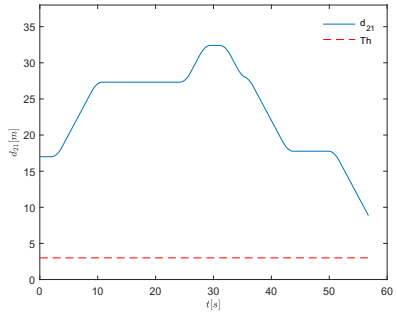


Figure 4.3: A simple AWS with three vehicles moving along a circuit.

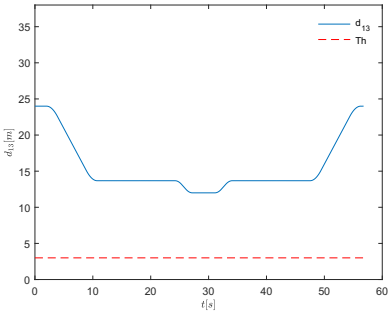
the first, the second, the third and the fourth zone respectively. In blue the place p_s used to connect the guide-path subsystem with the vehicle subsystem is shown.

4.3.3 Toy example

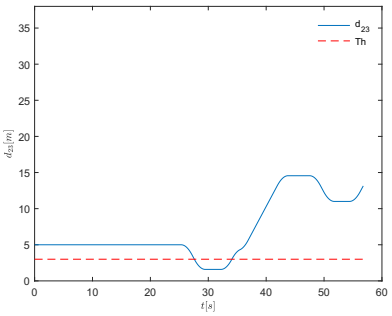
The simple system of Fig. 4.3 is modeled to show the effectiveness of CMHPNs in modeling AWSs. Thus, let V_1, V_2 and V_3 be three vehicles moving along a circuit and whose behaviour has been described in Sec. 2.2. The circuit is divided into four zones with $L_1 = L_3 = 8m$ and $L_2 = L_4 = 15m$. Each vehicle moves along the circuit and stops at a predefined position to perform a load/unload operation: after $\delta_{stop} = 5s$ it starts to move again. Notice that the distance between two consecutive vehicles must be always less than or equal to a fixed threshold $Th = 3m$. This whole system can be modeled using the net shown in Fig. 4.1, adding to it three colors and the subnet modeling the circuit. Results of the toy example simulation are reported in Fig. 4.4 and in Fig. 4.5. In detail, in Fig. 4.4a, Fig. 4.4b and Fig. 4.4c, the evolution of the relative distance between the vehicles is shown: d_{ij} is the distance from V_i to V_j, moving in a clockwise direction. Notice that only d_{23} goes



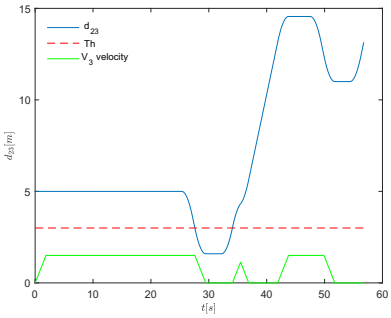
(a) Relative distance between V_1 and V_2.



(b) Relative distance between V_1 and V_3.



(c) Relative distance between V_2 and V_3.



(d) Evolution of V_3 velocity with respect to the relative distance between V_2 and V_3.

Figure 4.4: Results of the toy example simulation, with $L_1 = L_3 = 8m$ and $L_2 = L_4 = 15m$, $v_{max} = 1.5m/s$ and $a = 0.8m/s^2$.

under the threshold Th during the system evolution since V_3 stops to perform a load operation. In Fig. 4.4d, evolution of V_3 velocity with respect to the evolution of d_{23} is shown: to avoid collisions, velocity decreases each time d_{23} goes under the threshold value. In addition, notice that the velocity also decreases when a load (unload) operation must be performed (i.e when $t[s] > 50$).

In Fig. 4.5 the evolution of the position and the velocity of the three vehicles of the system has been reported for completeness. The position x_1 and the velocity x_2 of the vehicles are shown

as a solid blue line on top of which points of different colors are depicted: load and the unload operations are represented with yellow points and red points represent the acceleration/deceleration phases of vehicle due to its proximity to another one (stop-and go phenomena, more details are given in Sec. 5). Notice that V_2 must stop in order to avoid a collision and the acceleration phase takes place once the distance constraint is satisfied again. Also notice that all vehicles does not exceed their maximum velocity $v_{max} = 1.5m/s$.

4.4 Automatic model generation algorithm

The automatic synthesis procedure has been exploited to quickly set up different types of simulation configurations (see Sec.5). The conceived algorithm takes as input some high-level parameters related to the vehicle subsystem and the guide-path subsystem and allows the model of the whole IS to be automatically generated. Thanks to the adoption of this method, the time required for setting up a simulation was drastically reduced. Specifically, for the first subsystem there is the need to specify:

- the number of vehicles N_v;
- the initial position and velocity of each vehicle V_r, with $r \in \{1, \ldots, N_v\}$;
- the time δ_{stop} that each vehicle V_r spends to the load (unload) operation;
- the initial zone $Z_i, i \in \{1, \ldots, N_z\}$ of each vehicle V_r, with $r \in \{1, \ldots, N_v\}$.

For the second subsystem, the following parameters must be indicated:

- the number of zones N_z;

- the length L_i of each zone Z_i, with $i \in \{1, \ldots, N_z\}$;
- the topology $\mathbf{M}$ of the guide-path circuit.

As described in Sec. 4.3.1, in order to obtain the CMHPN model IS_{net} of the whole IS, the CMHPN model IS_v of the vehicle subsystem and the CMHPN model IS_g of the guide-path subsystem must be firstly created and then appropriately concatenated. After this, the setting of colors will implement the addition/removal of resources while remaining the net structure unchanged.

To obtain the CMHPN model IS_v, let P_v and T_v be the sets of :

- the places $\{p_s, p_{acc}, p_{const}, p_{dec}, p_{exch}, p_{dec\_stop}, p_{exch\_stop}\}$;
- the transitions $\{t_{acc}, t_{const}, t_{dec}, t_{exch}, t_i \ \forall i \in \{1, \ldots, 11\}\}$.

described in Sec. 4.3.1 and also represented in Fig. 4.1. Then, IS_v is obtained as follows:

1. for all $p_i \in P_v$ and $t_k \in T_v$, compute the sets $\mathcal{C}o(p_i)$ and $\mathcal{C}o(t_k)$ such that:

$$\begin{aligned} \mathcal{C}o(p_i) &= \{a_{i,1}, \ldots, a_{i,N_v}\} \quad = \\ \mathcal{C}o(t_k) &= \{a_{k,1}, \ldots, a_{k,N_v}\} \end{aligned}$$

   both of N_v colors (the number of colors of each place and each transition corresponds to the number of vehicles of the entire IS);

2. for the colored continuous place $p_s \in P_v$, define the structured markings $\boldsymbol{m}_{p_s} : \mathcal{C}o(p_s) \rightarrow (\mathbb{R}^+)^2$ as a vector of 2 non-negative real numbers (i.e. the marking of p_s with respect to the color r is $< x_1, x_2 >_r = \boldsymbol{x}_r$);

3. for each colored continuous transition $t_k \in T_v$ define the vector of firing speed $\boldsymbol{\nu} = (\nu_{k,1}, \ldots, \nu_{k,N_v})^T$ with the generic term $\nu_{k,r} = \boldsymbol{A} \cdot \boldsymbol{x}_r + \boldsymbol{B} \cdot a$;

4. for the colored discrete transition $t_{11} \in T_v$ define the firing delay vector $\boldsymbol{\delta}_{11} = (\delta_{11,1}, \ldots, \delta_{11,N_v})^T$. $\delta_{11,r}$ represents the time that the vehicle V_r takes to perform its load/unload operation; $\boldsymbol{\delta}_j = \mathbf{0}\ \forall j \in \{1, \ldots, 10\}$;

5. for each arc that connects the colored continuous place $p_s \in P_v$ with each colored continuous transition (and viceversa), define the appropriated block diagonal matrices of Eq. 4.5;

6. mark one of the colored discrete place $p_i \in IS_v$ with respect to the color r to assign the initial dynamic of V_r (also according to the values specified in Step 3).

In order to automatically generate the second model IS_g, a square matrix $\mathbf{M} \in \mathbb{R}^{N_z \times N_z}$ is used to represent the topology of the guide-path subsystem (i.e. how zones are connected to each other). In detail, the generic element (i, k) of $\mathbf{M}$ is:

$$M(i,k) = \begin{cases} 1 & \text{if the zone } i \text{ is connected with the zone} \\ & k \ \wedge \text{the driving direction is from the} \\ & \text{zone } i \text{ to the zone } k; \\ 0 & \text{otherwise} \end{cases}$$

Using such a representation, note that:

- the diagonal elements of $\mathbf{M}$ are all zero except when the IS is composed by only one zone;
- the total number of connections is represent by the sum of all the elements of $\mathbf{M}$ (i.e. $\sum_{i=1}^{N_Z} \sum_{k=1}^{N_Z} \mathbf{M}(i,k)$);
- $\mathbf{M}$ is not symmetric (due to the driving direction constraints);
- the presence of branch-points is dictated by the presence of more than one 1 on a row (or a column).

Thus, in order to obtain IS_g, let P_g be the set of N_Z places which represent the N_Z zones. Then:

1. for each element $M(i,k) = 1$, create a discrete transition $t_{i,k}$; let $T_g \subset T$ be the set of all the generated transitions (remember that such transitions will model the connection between the zone Z_i and the zone Z_k);

2. for each discrete transition $t_{i,k} \in T_g$ create an arc from the discrete place $p_i \in P_g$ to the transition $t_{i,k}$ and an arc from $t_{i,k}$ to the discrete place $p_k \in P_g$;

3. for all $p_i \in P_g$ and $t_{i,k} \in T_g$, compute the sets

$$\begin{aligned} \mathcal{C}o(p_i) &= \{a_{i,1}, \ldots, a_{i,N_v}\} \quad = \\ \mathcal{C}o(t_{i,k}) &= \{a_{\{i,k\},1}, \ldots, a_{\{i,k\},N_v}\} \end{aligned}$$

   both of N_v colors;

4. mark the place $p_i \in P_g$ with respect to the color r to assign the initial zone Z_i to the vehicle V_r.

5. for each arc that connects the colored continuous place $p_s \in P_v$ with each colored continuous transition $t_{i,k} \in T_g$, define the weights of Eq. 4.6;

4.4.1 Example of use

In this section, an example of use of the automated model generation algorithm is provided. To this aim, suppose the following high-level parameters for the vehicle subsystem:

- $N_v = 2$;
- $[2,0]^T$ and $[6,0]^T$ as position and velocity of the vehicle V_1 and V_2 respectively;
- $\delta_{stop} = 5s$;
- $Z_{i_1} = 1$ and $Z_{i_2} = 1$ as initial zone of the vehicle V_1 and V_2 respectively.

Moreover, suppose the following high-level parameters for the guide-path subsystem:

- $N_z = 4$;
- $L_1 = 10m$, $L_2 = 20m$, $L_3 = 30m$, $L_4 = 40m$;
- $$\mathbf{M} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Thus, the vehicle subsystem is composed by two vehicles which are in position 2 and position 6 of the zone 1. Furthermore, both of them have initial speed equal to zero and the time to perform the load (unload) operation is $5s$. The guide-path subsystem is composed by four zones which have length of $10m$, $20m$, $30m$ and $40m$, respectively. Moreover, as it is possible to understand from the matrix $\mathbf{M}$, the first zone is connected with the second and the third one, while these second two are connected only with the fourth zone. The last zone is connected to the first. The matrix $\mathbf{M}$ represents the topology of the guide-path subsystem of Fig. 4.2.

Starting from the above cited high-level parameters, IS_v is defined as follows:

1. each place $p_i \in P_v$ and each transition $t_k \in T_v$, has two colors: $\mathcal{C}o(p_i) = \mathcal{C}o(t_k) = \{a_1, a_2\}$;
2. the structured marking of the colored continuous place p_s with respect to the colors a_1 and a_2 is:
   - $p_{s_1} =< x_1, x_2 >_1 =< 2, 0 >= \boldsymbol{x}_1$;
   - $p_{s_2} =< x_1, x_2 >_2 =< 6, 0 >= \boldsymbol{x}_2$;
3. for each colored continuous transition $t_k \in T_v$ the firing speed vector is $\boldsymbol{\nu} = (\nu_{k,1}, \nu_{k,2})^T$ with:
   - $\nu_{k,1} = \boldsymbol{A} \cdot \boldsymbol{x}_1 + \boldsymbol{B} \cdot a$;

- $\nu_{k,2} = \boldsymbol{A} \cdot \boldsymbol{x}_2 + \boldsymbol{B} \cdot a$;

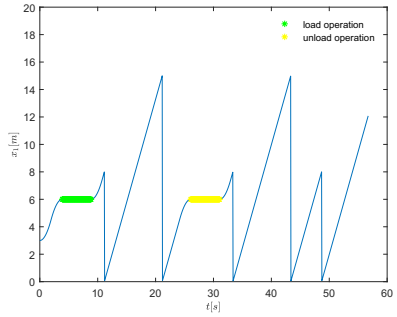
4. for the colored discrete transition t_{11} the firing delay vector is $\boldsymbol{\delta}_{11} = (\delta_{11,1}, \delta_{11,2})^T$ with:
   - $\delta_{11,1} = \delta_{stop} = 5$;
   - $\delta_{11,2} = \delta_{stop} = 5$;
5. the matrices of weights of each arc that connect $p_s \in P_v$ with t_{acc}, t_{const}, t_{dec}, t_{exch} (and viceversa) are:

$$\mathbf{Post}(p_s, t_{acc}) = \begin{bmatrix} \boldsymbol{\pi}_3 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\pi}_3 \end{bmatrix}$$
$$\mathbf{Pre}(p_s, t_{acc}) = \begin{bmatrix} \boldsymbol{\pi}_2 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\pi}_2 \end{bmatrix}$$
$$\mathbf{Post}(p_s, t_{const}) = \begin{bmatrix} I_2 & \mathbf{0} \\ \mathbf{0} & I_2 \end{bmatrix}$$
$$\mathbf{Pre}(p_s, t_{const}) = \begin{bmatrix} \boldsymbol{\pi}_2 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\pi}_2 \end{bmatrix}$$
$$\mathbf{Post}(p_s, t_{dec}) = \begin{bmatrix} \boldsymbol{\pi}_1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\pi}_1 \end{bmatrix}$$
$$\mathbf{Pre}(p_s, t_{dec}) = \begin{bmatrix} \boldsymbol{\pi}_2 & 0 \\ 0 & \boldsymbol{\pi}_2 \end{bmatrix}$$
$$\mathbf{Post}(p_s, t_{exch}) = \begin{bmatrix} \boldsymbol{\pi}_1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\pi}_1 \end{bmatrix}$$
$$\mathbf{Post}(p_s, t_{exch}) = \begin{bmatrix} \boldsymbol{\pi}_2 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\pi}_2 \end{bmatrix}$$

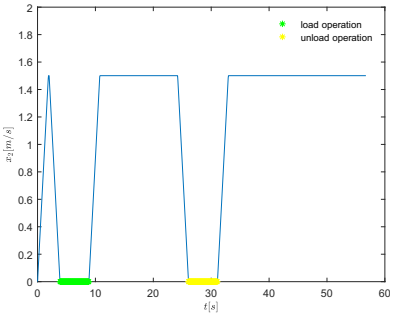
   where $\mathbf{0}$ is a 2×2 null matrix;
6. the marking of the place p_{acc} with respect to the color a_1 and a_2 is c_1 and c_2 respectively. The other colored discrete places are not marked.

Concerning the guide-path subsystem, let $P_g = \{p_1, p_2, p_3, p_4\}$ be the set of four places which represent the four zones. Then:

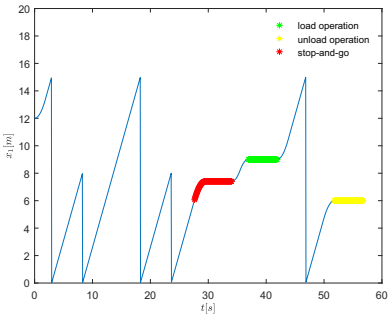
1. the discrete transition $t_{1,2}$, $t_{1,3}$, $t_{2,4}$, $t_{3,4}$ and $t_{4,1}$ are created;
2. the preset and the postset of the each previously created transitions are generated:
   - ${}^\bullet t_{1,2} = \{p_1\}$ and $t_{1,2}^\bullet = \{p_2\}$;
   - ${}^\bullet t_{1,3} = \{p_1\}$ and $t_{1,3}^\bullet = \{p_3\}$;
   - ${}^\bullet t_{2,4} = \{p_2\}$ and $t_{2,4}^\bullet = \{p_4\}$;
   - ${}^\bullet t_{3,4} = \{p_3\}$ and $t_{3,4}^\bullet = \{p_4\}$;
   - ${}^\bullet t_{4,1} = \{p_4\}$ and $t_{4,1}^\bullet = \{p_1\}$;
3. each place p_i and each transition $t_{i,k}$ has two colors:
$$\mathcal{C}o(p_i) = \mathcal{C}o(t_{i,k}) = \{a_1, a_2\};$$
4. the marking of the place p_1 with respect to the colors a_1 and a_2 is c_1 and c_2 respectively; the other places are not marked;
5. the matrices of weights of each arc that connect $p_s \in P_v$ with transition $t_{i,k}$ are:
$$\mathbf{Pre}(p_s, t_{1,2}) = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} = \mathbf{Pre}(p_s, t_{1,3})$$
$$\mathbf{Pre}(p_s, t_{2,4}) = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$$
$$\mathbf{Pre}(p_s, t_{3,4}) = \begin{bmatrix} 30 & 0 \\ 0 & 30 \end{bmatrix}$$
$$\mathbf{Pre}(p_s, t_{4,1}) = \begin{bmatrix} 40 & 0 \\ 0 & 40 \end{bmatrix}$$



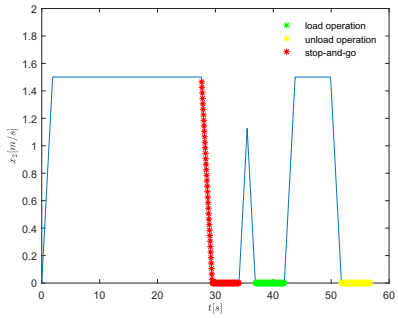
(a) Evolution of the position x_1 of the vehicle V_1.



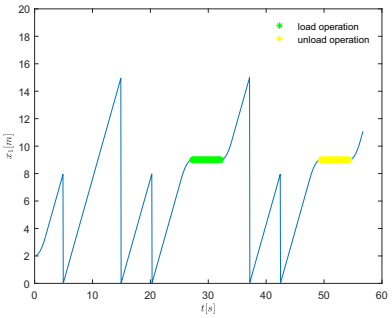
(b) Evolution of the velocity x_2 of the vehicle V_1.



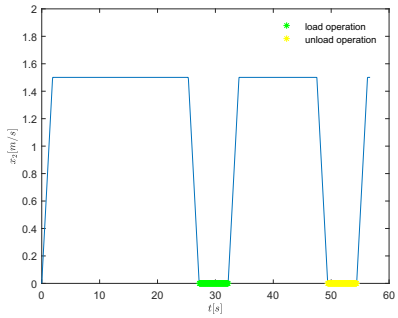
(c) Evolution of the position x_1 of the vehicle V_2.



(d) Evolution of the velocity x_2 of the vehicle V_2.



(e) Evolution of the position x_1 of the vehicle V_3.



(f) Evolution of the velocity x_2 of the vehicle V_3.

Figure 4.5: Evolution of the position x_1 and the velocity x_2 of the vehicles shown as a solid blue line on top of which points of different colors are shown: load and the unload operations are represented with yellow points and red points represent stop-and go phenomena.

Chapter 5

Case study

In this section, details about the performed simulations are provided and the problem of tuning appropriately the auction-based algorithm is addressed. In detail, Sec. 5.1 reports the results of the conducted experiments with a simulator based on the core of PNetLab, a simulation and analysis tool developed by the Automatic Control Group of the University of Salerno (Università degli Studi di Salerno) [55]. The simulator takes as input a CMHPN (described using an XML formalism) and allows the controller to be implemented as a C/C++ program. In Sec. 5.2, it is shown that, on the basis of a detailed simulation study, where the set of fully dimensionless parameters common to any AWS discussed in Sec.3 is considered, a design methodology can be devised as follows: the set of dimensionless parameters can be set to starting values guaranteeing acceptable performance, and then, using a more accurate simulation study on the specific case, a finest tuning can be achieved.

5.1 Simulations

The automatic model synthesis procedure discussed in Sec.4.4 was used to quickly set-up several types of simulation configurations. Each configuration is identified by the number of vehicles, the state of each them, their load/unload operation time, their ini-

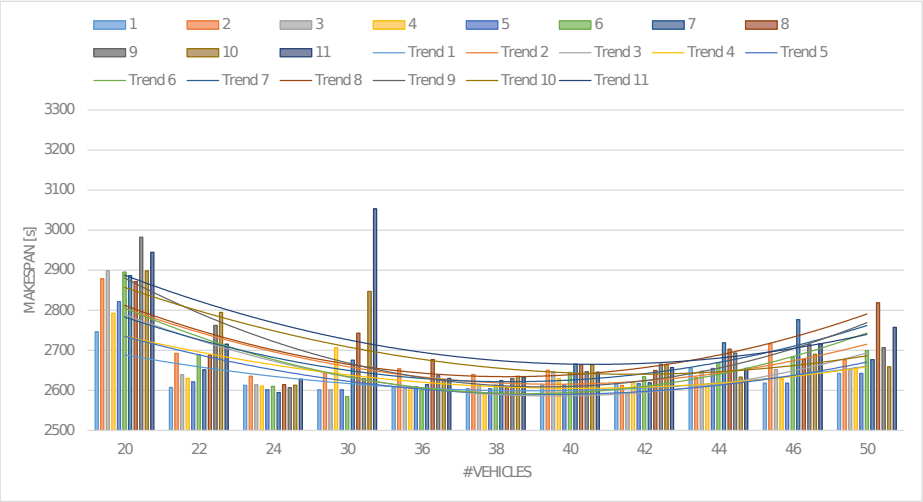


Figure 5.1: Makespan obtained on the 500 mission dataset while varying the number of vehicles and considering the combination of heuristics reported in Tab. 5.2.

tial zone, the layout of the IS and the control laws to be tested. A total of 363 simulations have been executed in order to evaluate the performance of the centralized control architecture and the auction-based one. Notice that each configuration requires the generation of a new model. Performing such a high number of simulations could have been onerous from a time perspective without the use of the automatic model synthesis algorithm.

The guide-path subsystem of the used IS is made up of 15 zones of length $25m$, $5m$, $25m$, $24m$, $24m$, $4m$, $9m$, $25m$, $5m$, $25m$, $9m$, $24m$, $24m$, $4m$, $55m$ respectively. In the first and in the third zone, 4 picking bays are present while in the 8-th and in the 10-th zone stand 4 deposit bays. The distance between the first two and the last two bayes in each zone is $6m$ and the distance between the second and the third one is $5m$. The connection between the zones is reported in Fig. 2.3. Moreover, for the auction-based algorithm, experiments have also been performed by varying the parameters τ_s and τ_r in the set $\{0.2, 0.4\}$ to assume a reliable communication among all the entities and $\tau_{bid} = 1ms$. Furthermore, as

	1	2	3	4	5	6	7	8	9	10	11
20	2745,91	2878,35	2898,38	2792,74	2821,23	2895,49	2886,57	2871,62	2982,37	2898,02	2944,6
22	2607,42	2692,39	2638,54	2630,18	2620,95	2688,8	2651,14	2688,29	2762,28	2794,68	2715,49
24	2612,69	2635,35	2613,67	2610,85	2601,32	2610,15	2594,55	2614,11	2606,7	2612,91	2628,1
30	2600,92	2642,52	2601,17	2705,8	2600,92	2584,69	2675,45	2742,88	2628,55	2847,1	3052,9
36	2608,88	2653,83	2604,22	2609,56	2608,88	2600,56	2614,33	2676,94	2636,69	2627,31	2629,85
38	2603,72	2639,71	2612,23	2594,22	2603,72	2613,04	2624,44	2607,06	2628,92	2635,12	2633,23
40	2613,4	2650,21	2647,12	2629,33	2613,4	2644,82	2665,68	2664,04	2646,69	2663,42	2645,38
42	2616,87	2612,33	2589,72	2621,5	2616,87	2633,85	2618,63	2649,66	2665,38	2665,33	2656,92
44	2654,75	2632,3	2647,7	2609,44	2654,75	2669,22	2718,66	2702,87	2693,24	2633,17	2654,91
46	2618,17	2716,42	2651,92	2628,47	2618,17	2682,97	2776,72	2678	2714,3	2690,2	2716,14
50	2642,34	2676,34	2651,6	2656,12	2642,34	2699,41	2676,8	2818,42	2706,59	2658,57	2757,31

Table 5.1: Makespan obtained on the 500 mission dataset while varying the number of vehicles and considering the combination of heuristics reported in Tab. 5.2.

for the vehicles and their related constraints it is: $v_{max} = 1.5m/s$, $a = 0.8m/s^2$, $\delta_{stop} = 5s$ and $Th = 3m$. The maximum number of forward simulations to execute is set to $N_{max\_fw} = 3$. Indeed, because the mean duration of a simulation is $5s$ and the mean duration to complete a mission is $140s$, this setting allows to complete the forward simulation in $15s$. It is assumed that each vehicle

#	**Approach**	H_{who}	H_{way}	$\tau_s(s)$	$\tau_r(s)$
1	Centralized	H_{who_1}	H_{way_1}		
2	Centralized	H_{who_2}	H_{way_2}		
3	Centralized	H_{who_1}	H_{way_2}		
4	Centralized	H_{who_2}	H_{way_1}		
5	Centralized	H_{who_4}	H_{way_1}		
6	Auction-based	H_{who_1}	H_{way_1}	0.2	0.2
7	Auction-based	H_{who_1}	H_{way_1}	0.2	0.4
8	Auction-based	H_{who_1}	H_{way_1}	0.4	0.2
9	Auction-based	H_{who_3}	H_{way_1}	0.2	0.2
10	Auction-based	H_{who_3}	H_{way_1}	0.2	0.4
11	Auction-based	H_{who_3}	H_{way_1}	0.4	0.2

Table 5.2: Combination of heuristics used in each simulation.

can execute only one mission at a time and each mission requires exactly one vehicle in order to be executed.

A set of missions is given as input to the IS in an asynchronous manner, namely, datasets composed by 500, 600 and 700 uniformly distributed missions in the time intervals $[0, 2500]$, $[0, 3000]$ and $[0, 3500]$ respectively.

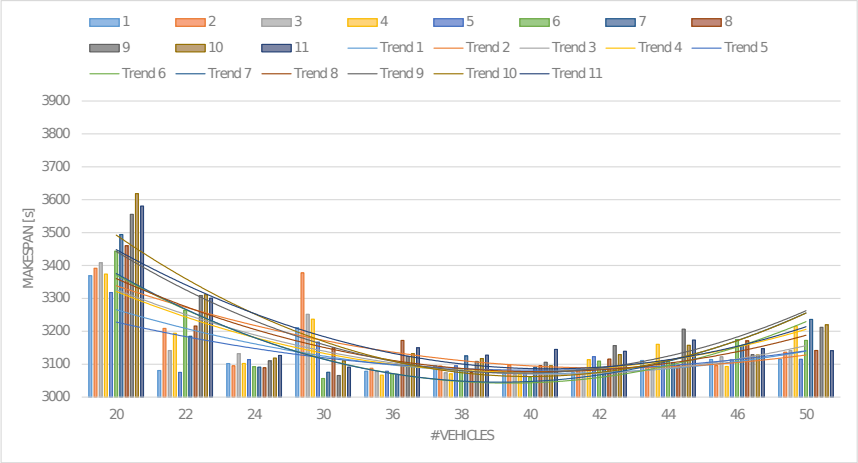


Figure 5.2: Makespan obtained on the 600 mission dataset while varying the number of vehicles and considering the combination of heuristics reported in Tab. 5.2.

The results obtained using the datasets described above are summarized in Fig. 5.1, Fig. 5.2 and Fig. 5.4 respectively.

In such figures, the makespan of each simulation with a variable number of vehicles is shown. The complete list of simulations is reported in Tab.5.2. In detail, for each vehicle, the makespan of each simulation is reported as a colored vertical bar and its trend (among all the performed simulations) has been shown as a line of the same color. For example, the light red vertical bar represents the makespan obtained in each simulation where the centralized approach is employed, $H_{who} = H_{who_2}$ and $H_{way} = H_{way_2}$. Rather, the light red line represents the trend of the maskespan with the just described parameters while varying the number of vehicles. Moreover, in Tab. 5.1, Tab. 5.3 and Tab. 5.4, the makespan of each simulation while varying the number of vehicles is also numerically reported for a better understanding of the experiments. From the

	1	2	3	4	5	6	7	8	9	10	11
20	3368,93	3391,72	3408,33	3374,03	3317,84	3443,95	3494,14	3460,1	3555,62	3618,4	3580,82
22	3080,82	3208,83	3141,58	3194,11	3075,13	3263,9	3185,06	3216,12	3308,86	3311,04	3300,67
24	3102,04	3095,05	3132	3102,27	3114,01	3092,3	3091,15	3089,16	3110,06	3118,77	3126,66
30	3210,2	3377,61	3251,84	3236,9	3167,41	3056,58	3075,55	3149,53	3065,66	3110,15	3091,28
36	3078,68	3087,69	3077,55	3066,83	3078,68	3071,33	3068,99	3172,29	3123,92	3131,8	3149,87
38	3094,82	3094,48	3074,05	3070,48	3094,82	3082,27	3125,06	3075	3108,48	3117,18	3127,69
40	3072,51	3096,89	3068,7	3069,47	3072,51	3059,23	3090,7	3095,36	3105,96	3094,87	3144,99
42	3073,45	3083,02	3084,3	3113,76	3122,77	3109,43	3083,8	3115,73	3156,57	3129,19	3139,46
44	3111,3	3092,43	3093,21	3160,46	3111,3	3110,22	3105,24	3096,22	3206,66	3157,6	3173,55
46	3114,2	3095,79	3122,37	3093,13	3114,2	3174,97	3156,8	3171,01	3129,07	3128	3147,13
50	3115,08	3136,55	3141,23	3214,98	3115,08	3172,41	3236,15	3141,74	3212,14	3219,82	3141,07

Table 5.3: Makespan obtained on the 600 mission dataset while varying the number of vehicles and considering the combination of heuristics reported in Tab. 5.2.

above cited figures it is evident that, at the beginning, as the number of vehicles increases, the makespan decreases. However, as the number of vehicles continues to increase, the makespan also begins to grow up. This is mainly due to the so called "stop and go" phenomena. Indeed, when the number of vehicles increases, it very frequently happens that some vehicles have to decelerate because they are too close to each other or, sometimes, they have to stop. For example, in Fig. 5.3, the behaviour of the first vehicle on the first dataset when $H_{who} = H_{who_1}$ and $H_{way} = H_{way_1}$ is reported. In such a figure, the position x_1 and the velocity x_2 of the vehicle are shown as a solid blue line, on top of which points of different colors are shown. In detail, the load and the unload operations are represented with yellow points and red points represent the acceleration/deceleration phases of a vehicle related to the stop and go phenomena. The vehicle decelerates in the proximity of another one and, sometimes, it must stop. The acceleration phase takes place once the distance constraint is satisfied again.

Note that, with the used hybrid model, it has been possible to evaluate different control approaches in the employed IS, showing how stop and go phenomena and the usage of different heuristics affect the system performance. For example, in the first dataset, it is evident that the system performance improves at the migration from 30 to 36 vehicles. This is also true for the second and the

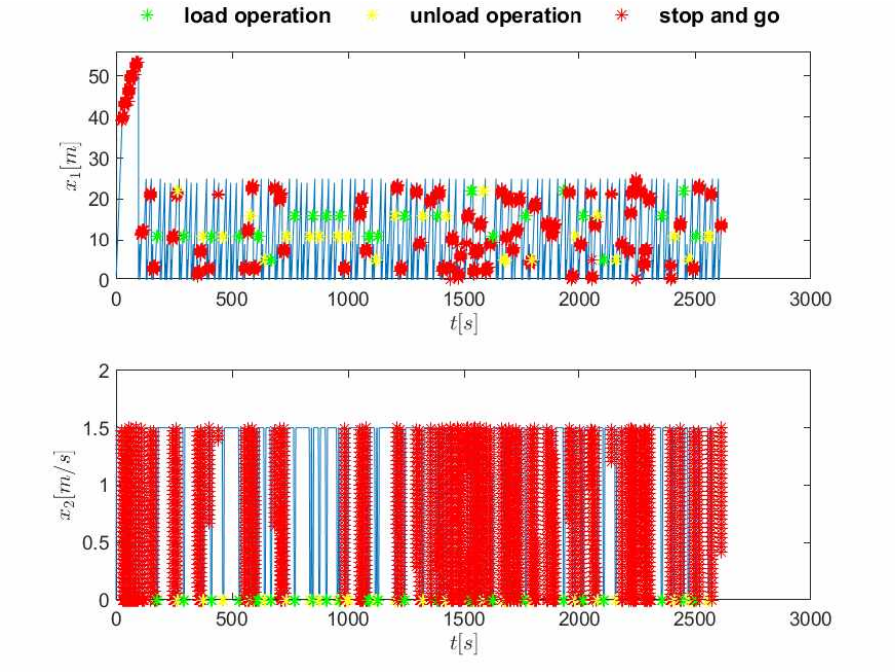


Figure 5.3: Position x_1 and the velocity x_2 of the vehicle number one of the first dataset shown as blue lines. On their top, points representing the load and the unload operations and acceleration/deceleration phases (related to the stop and go phenomena) are shown in green, yellow and red respectively.

third dataset. Concerning the heuristics, it seems evident that H_{who_3} in the auction-based approach should be avoided as it yields an higher makespan than other employed rules.

Furthermore, it may result that the makespan of some simulations, where the auction-based approach is used, is lower than that obtained when the centralized approach is employed. This might be due to many factors. Suppose, for example, that the state of the IS in the simulation in which the centralized task allocation algorithm is used, is the same of the simulation where the auction-based approach is employed. Moreover, suppose that a mission m must be allocated: in the first simulation this oper-

	1	2	3	4	5	6	7	8	9	10	11
20	3841,56	3977,83	3945,87	3870,91	3852,21	4063,34	4010,32	4020,98	4132	4261,4	4141,99
22	3609,67	3679,46	3686,58	3654,87	3628,32	3707,11	3676,82	3703,77	3847,44	3885,86	3800,33
24	3602,7	3683,61	3624,55	3637,58	3600,9	3655,52	3646,19	3630,07	3626,09	3641,81	3716,56
30	3691,09	3820,6	3786,51	3846,44	3741,34	3767,44	3815,69	3768,69	3906,09	3607,86	3580,76
36	3571,8	3595,98	3587,29	3575,99	3571,8	3640,23	3614,94	3646,94	3647,59	3624,46	3612,63
38	3575,94	3595,21	3589,79	3618,85	3575,94	3577	3595,87	3583,03	3611,91	3681,44	3626,14
40	3594,73	3612,59	3581,63	3601,26	3594,73	3639,51	3579,71	3621,07	3614,98	3629,27	3671
42	3626,4	3637,27	3611,45	3592,49	3633,53	3590,56	3641,16	3629,99	3644,4	3654,5	3627,68
44	3643,83	3618,5	3666,57	3596,21	3643,83	3644,27	3643,44	3638,52	3649,8	3657,58	3623,78
46	3653,09	3652,69	3687,1	3637,36	3653,09	3656,83	3660,49	3638,2	3720,41	3677,67	3766,66
50	3658,47	3628,61	3663,96	3611,04	3658,47	3715,17	3650,45	3679,08	3685,99	3788,91	3696,38

Table 5.4: Makespan obtained on the 700 mission dataset while varying the number of vehicles and considering the combination of heuristics reported in Tab. 5.2.

ation is instantaneously performed; in the second case, the same allocation is delayed. It may happen that the vehicle chosen in the first case releases its assigned SU before the vehicle of the second simulation does it. However, when a new mission arrives, the second vehicle may be in a position where it is faster to reach the picking bay and to complete the mission right away. Obviously, this is not the only phenomenon that can occur. In addition, as it is evident from the performed experiments, only in few cases the auction-based approach results to be better than the centralized one and in the most cases the performance of the auction-based method are at most 13% worse with respect to the centralized approach. However, even if it is evident the makespan obtained with the auction-based approach can be worse than the one obtained to a centralized one, this occurs mainly when the number of vehicles is far from the best operating points (configuration from 36 to 42 vehicles). Indeed, for these configurations the difference is often negligible (less than 1%) and sometimes auction-based solutions leads to better results.

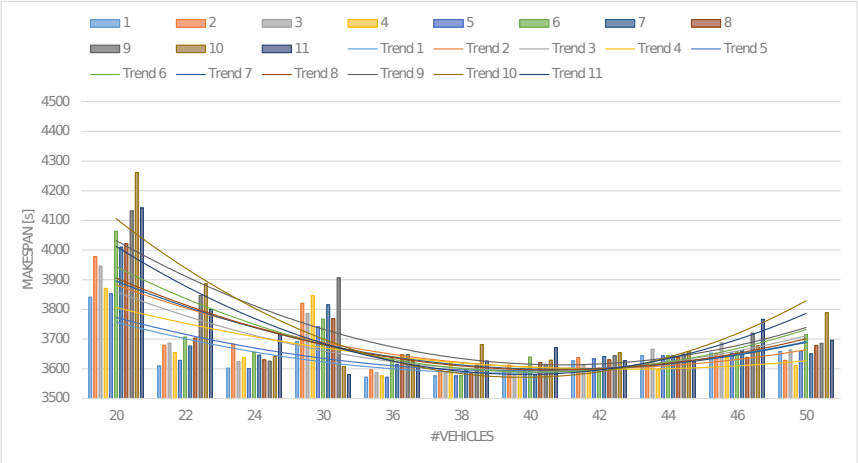


Figure 5.4: Makespan obtained on the 700 mission dataset while varying the number of vehicles and considering the combination of heuristics reported in Tab. 5.2.

5.2 Toward a design methodology for auction-based algorithms

The problem of tuning appropriately the auction-based algorithm is challenging. In this section, on the basis of a detailed simulation study, where the set of fully dimensionless parameters common to any AWS discussed in Sec.3 is considered, it is shown that a design methodology can be devised as follows: the set of dimensionless parameters can be set to starting values guaranteeing acceptable performance, and then, using a more accurate simulation study on the specific case, a finest tuning can be achieved. To this aim, several types of simulation configurations have been exploited considering $H_{who} = H_{who_{1b}}$ and $H_{way} = H_{way_1}$. The complete list of simulation configurations used to compare the performance of the different analysed auction types is reported in Tab.5.5.

In the first configuration, only one mission is auctioned at a time (i.e. $b_s = 1$). In addition, all the vehicles participate to the auction (i.e. $r_p = L_{max}$) and the closest vehicle to the withdrawal

#	k	r_p	b_s
[1 - 4]	1	L_{max}	$[1-4]$
[5 - 6]	1	$[20-40]$	1
[7 - 10]	$[1-4]$	L_{max}	1

Table 5.5: List of simulation configurations used to compare the performance of different auction types in the case study: k and r_p permit to select the k-th closest vehicle at distance r_p from the exchange point where a SU must be withdrawn; b_s is the number of missions auctioned.

bay of the mission $m \in M_{list}$ is chosen (i.e. $k = 1$). In the subsequent three configurations, the parameters r_p and k are the same of the former one and, $b_s \in \{2, 3, 4\}$. In the 5-th and 6-th configuration, a distance of $r_p = 20m$ and $r_p = 40m$ is considered while remaining unchanged the other two parameters, both set equal to one. In the last four simulation configurations, a value of $k \in \{1, 2, 3, 4\}$ is considered with $r_p = L_{max}$ and $b_s = 1$.

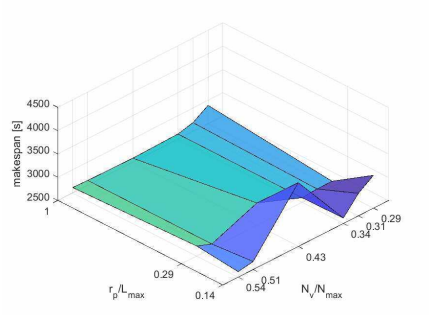
As for the other simulation parameters we have: $N_{max} = 70$, $\tau_{bid} = 1ms$ and $\tau_s = \tau_r = 0.2s$. The sets of 500, 600 and 700 uniformly distributed missions described in the previous section are given as input to the IS in an asynchronous manner respectively.

In Fig. 5.5, Fig. 5.6 and Fig. 5.7 the makespan obtained using the datasets described in the above section while varying the dimensionless value N_v/N_{max} together with dimensionless value r_p/L_{max}, k and b_s is reported. Notice that, when $N_v/N_{max} > 0.5$, the makespan starts to increase. As also descibed in the previous section, this is due to the so called stop-and-go phenomena: vehicles are involved in frequent acceleration/deceleration phases and, often, they have to stop [36]. The frequent presence of stop-and-go phenomena is a clear signal that the system is saturated. Concerning r_p/L_{max}, a maximum value of approximately 0.3 guarantees good results: at a cost a negligible price, communicating only with a subset of vehicles can help to reduce computational and communication issues. Regarding the parameter k, it results

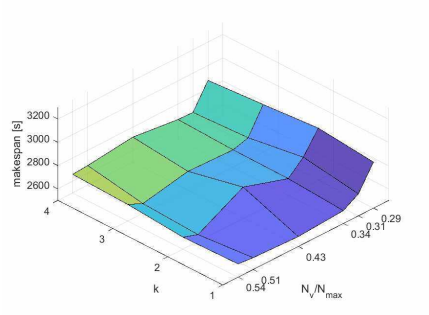
that: 1) the choice of k becomes irrelevant, when the system is saturated; 2) the accurate tuning of such a parameter permits to obtain an improvement of system performance, in the other cases. As for the parameter b_s, it is possible to observe that, when it increases, worst performance are obtained: decisions about the missions to perform are taken too early respect to their initial time. However, as the number of missions auctioned increases, the system's performance does not drop dramatically.

Lastly, in order to clearly summarize the obtained results, in Tab. 5.6 the minimum makepan for each simulation configuration is reported. Following the simulation results and considering the theoretical/implementation aspects of the proposed method (reported in Sec. 2, Sec. 3, and Sec. 4), it is remarkable to state that the deployed control architecture gives an unprecedented amount of flexibility to logistic systems: the employed heuristics-based control lows together with the accurate tuning of the auction parameters help to create high-performance and reliable solutions.

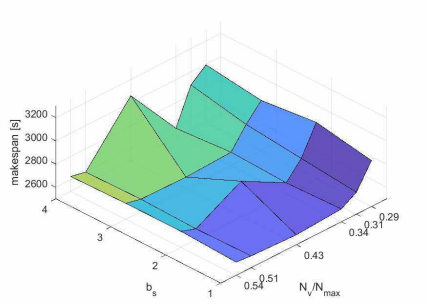
.



(a) Makespan obtained while varying N_v/N_{max} together with r_p/L_{max}.

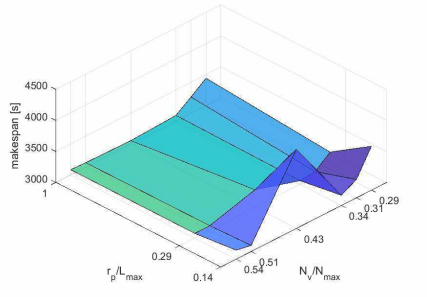


(b) Makespan obtained while varying N_v/N_{max} together with k.

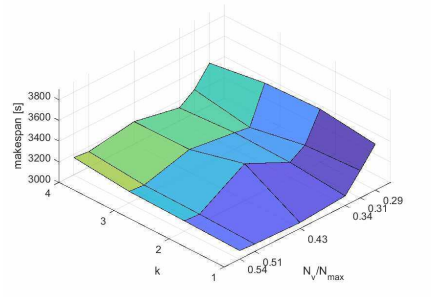


(c) Makespan obtained while varying N_v/N_{max} together with b_s.

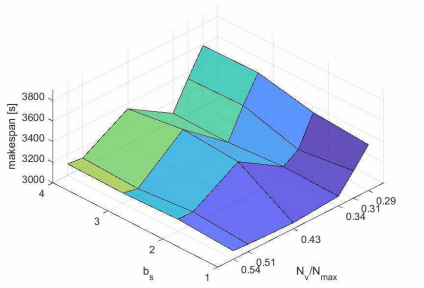
Figure 5.5: Makespan obtained with the dataset composed by 500 uniformly distributed missions in the time interval $[0, 2500]$ while varying N_v/N_{max} together with r_p/L_{max}, k and b_s when $\tau_s = \tau_r = 0.2s$ and $\tau_{bid} = 1ms$.



(a) Makespan obtained while varying N_v/N_{max} together with r_p/L_{max}.

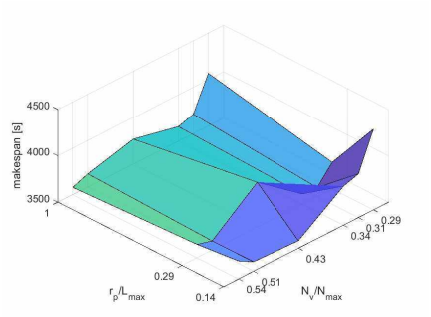


(b) Makespan obtained while varying N_v/N_{max} together with k.

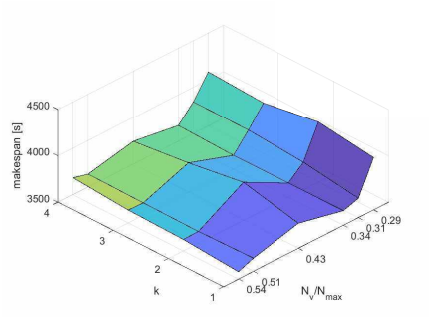


(c) Makespan obtained while varying N_v/N_{max} together with b_s.

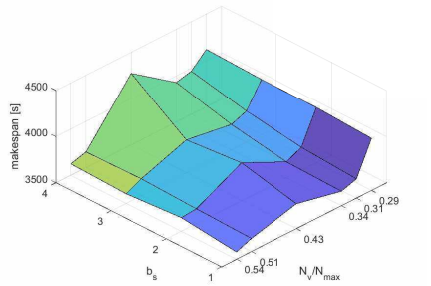
Figure 5.6: Makespan obtained with the dataset composed by 600 uniformly distributed missions in the time interval $[0, 3000]$ while varying N_v/N_{max} together with r_p/L_{max}, k and b_s when $\tau_s = \tau_r = 0.2s$ and $\tau_{bid} = 1ms$.



(a) Makespan obtained while varying N_v/N_{max} together with r_p/L_{max}.



(b) Makespan obtained while varying N_v/N_{max} together with k.



(c) Makespan obtained while varying N_v/N_{max} together with b_s.

Figure 5.7: Makespan obtained with the dataset composed by 700 uniformly distributed missions in the time interval $[0, 3500]$ while varying N_v/N_{max} together with r_p/L_{max}, k and b_s when $\tau_s = \tau_r = 0.2s$ and $\tau_{bid} = 1ms$.

#	Approach	H_{Who}	H_{Way}	k	$r_p(m)$	b_s	$\tau_s(s)$	$\tau_r(s)$	min. makespan (s) (500 mission dataset)	min. makespan (s) (600 mission dataset)	min. makespan (s) (700 mission dataset)
1	Centralized	H_{who_1}	H_{way_1}	1	L_{max}	1			2600,92	3072,51	3571,8
2	Centralized	H_{who_2}	H_{way_2}	1	L_{max}	1			2612,33	3083,02	3595,21
3	Centralized	H_{who_1}	H_{way_2}	1	L_{max}	1			2589,72	3068,7	3581,63
4	Centralized	H_{who_2}	H_{way_1}	1	L_{max}	1			2594,22	3069,47	3575,99
5	Centralized	H_{who_4}	H_{way_1}	1	L_{max}	1			2600,92	3072,51	3571,8
6	Auction-based	H_{who_1}	H_{way_1}	1	L_{max}	1	0,2	0,2	2581,69	3056,58	3577,0
7	Auction-based	H_{who_1}	H_{way_1}	1	L_{max}	1	0.2	0,4	2594,65	3068.99	3579,71
8	Auction-based	H_{who_1}	H_{way_1}	1	L_{max}	1	0,4	0,2	2607,06	3075	3583,03
9	Auction-based	H_{who_3}	H_{way_1}	1	L_{max}	1	0,2	0,2	2606,7	3065,66	3611,91
10	Auction-based	H_{who_3}	H_{way_1}	1	L_{max}	1	0,2	0,4	2612,91	3094,87	3607,86
11	Auction-based	H_{who_3}	H_{way_1}	1	L_{max}	1	0,4	0,2	2628,1	3091,28	3580,76
13	Auction-based	H_{who_1}	H_{way_1}	1	L_{max}	2	0,2	0,2	2595,95	3110,2	3647,52
14	Auction-based	H_{who_1}	H_{way_1}	1	L_{max}	3	0,2	0,2	2601,5	3073,54	3603,8
15	Auction-based	H_{who_1}	H_{way_1}	1	L_{max}	4	0,2	0,2	2582,43	3082,61	3624,04
16	Auction-based	H_{who_1}	H_{way_1}	1	20	1	0,2	0,2	2618,43	3098,57	3581,93
17	Auction-based	H_{who_1}	H_{way_1}	1	40	1	0,2	0,2	2595.4	3061,35	3597,77
19	Auction-based	H_{who_1}	H_{way_1}	2	L_{max}	1	0,2	0,2	2559,55	3090,73	3623,64
20	Auction-based	H_{who_1}	H_{way_1}	3	L_{max}	1	0,2	0,2	2585,06	3100.62	3632.44
21	Auction-based	H_{who_1}	H_{way_1}	4	L_{max}	1	0,2	0,2	2660,2	3130,96	3640.22

Table 5.6: Minimum makespan for each simulation configuration.

Chapter 6

Conclusions

In the field of logistics, the study of systems that follow the recommended requirements of the "Industry 4.0" is now of paramount importance. Since traditional hierarchical and centralized control architectures have been employed to solve most of the basic problems in this context, many researchers have shifted their focus to the study of more flexible ones. This is mainly due to the fact that nowadays vehicles can be equipped with high-performance, low-power and cheap electronic components that provide communication capabilities and processing power: they are able to work on their own and interact with each other exchanging messages. In the studied application domain, the main contribution of this dissertation concerned the development of an auction-based control architecture for the control of modern AWSs, characterized by a big IS composed by vehicles which can move SUs along a monodimensional guide-path. In detail, the tackled control problem focused on the assignment of each available vehicle to a mission without regarding the coordination of the entire fleet. Indeed, each of them is able to adjust their own velocity to avoid collisions since obstacles detection using simple low-cost sensors can be performed. Thus, vehicles can safely operate in structured environments, stopping the execution of a certain task if a potential risk is detected.

With regard to the examined control problem, it involved de-

termining which vehicle ("*Who*") must perform a certain mission ("*What*") and the identification of the best strategy to accomplish it ("*Which Way*") while minimizing the makespan, defined as the time to complete a set of missions. To this aim, theories and techniques borrowed by the robotic community were deeply studied: auctions have been widely analyzed and then adapted to the logistic field considering all the relevant constraints of this application domain. Concerning the latter point, since the response time to assign a mission makes the use of multi-round auctions unattractive due to their demanding computation/communication requirements, variants of one-shot k-price sealed-bid auctions have been designed and evaluated.

To face with the problem of the system performance evaluation and analysis and to support the development of auction-based control algorithms, a highly modular and compact hybrid model based on CMHPNs together with heuristic based control laws was designed. In addition, an automated model synthesis procedure was also developed, since the manual creation of the provided model can be a very demanding activity. The conceived model results highly modular (due to the identification of elementary modules of AWSs that can be concatenated to obtain different typologies of ISs) and compact (thanks to the introduction of colors and structured markings) while faithfully able to represent the most relevant dynamics of AWSs. In addition, with the automatic synthesis procedure, the addition/removal of resources as well as the set-up of simulations under different system configurations can be easily achieved.

To remedy the lack of design methods, that is still a problem for algorithms based on auction mechanisms, a design methodology for such algorithms was also devised: results to the problem of design/tuning appropriately the proposed auction-based control algorithm were also presented. Indeed, such algorithms can be firstly set-up with respect to a set of dimensionless parameters, and then, using a more accurate simulation study on the specific case, a finest tuning can be achieved.

Simulation results encourage the adoption of auction-based

control architectures in logistic environments as effective and efficient solutions. At the same time, modularity, real-time capabilities, robustness, services orientation capabilities and scalability are delivered to the whole system. It is worth remarking that computational and communication requirements, necessary to use the proposed auction-based approach, do not represent a drawback for AWSs due to the constraints of the logistic application domains.

Future research will focus also on the application to urban traffic system [56] of the approach presented in this work.

Bibliography

[1] N. Boysen, R. de Koster, and F. Weidinger, "Warehousing in the e-commerce era: A survey," *European Journal of Operational Research*, pp. 1–16, 2018.

[2] R. De Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *European journal of operational research*, vol. 182, no. 2, pp. 481–501, 2007.

[3] J. Clement, "Annual retail e-commerce sales growth worldwide from 2014 to 2020," Statista, Tech. Rep., 2019.

[4] F. Basile, P. Chiacchio, and J. Coppola, "A hybrid model of complex automated warehouse systemspart I: Modeling and simulation," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 4, pp. 640–653, Oct 2012.

[5] B. Rouwenhorst, B. Reuter, V. Stockrahm, G.-J. van Houtum, R. Mantel, and W. H. Zijm, "Warehouse design and control: Framework and literature review," *European journal of operational research*, vol. 122, no. 3, pp. 515–533, 2000.

[6] N. Boysen, S. Fedtke, and F. Weidinger, "Optimizing automated sorting in warehouses: The minimum order spread sequencing problem," *European Journal of Operational Research*, vol. 270, no. 1, pp. 386 – 400, 2018.

[7] N. Zaerpour, Y. Yu, and R. de Koster, "Small is beautiful: A framework for evaluating and optimizing live-cube compact storage systems," *Transportation Science*, vol. 51, no. 1, pp. 34–51, 2017.

[8] K. R. Gue and B. S. Kim, "Puzzle-based storage systems," *Naval Research Logistics (NRL)*, vol. 54, no. 5, pp. 556–567, 2007.

[9] K. Azadeh, R. De Koster, and D. Roy, "Robotized warehouse systems: Developments and research opportunities," *SSRN Electronic Journal*, 01 2017.

[10] A. C. Caputo and P. M. Pelagagge, "Management criteria of automated order picking systems in high-rotation high-volume distribution centers," *Industrial Management & Data Systems*, vol. 106, no. 9, pp. 1359–1383, 2006.

[11] R. Andriansyah, L. F. P. Etman, I. J. B. F. Adan, and J. E. Rooda, "Design and analysis of an automated order-picking workstation," *Journal of Simulation*, vol. 8, no. 2, pp. 151–163, May 2014.

[12] B. P. Gerkey and M. J. Mataric, "A framework for studying multi-robot task allocation," *Multi-robot Systems: From Swarms to Intelligent Automata*, 2003.

[13] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[14] F. Basile, P. Chiacchio, and J. Coppola, "A Hybrid Model of Complex Automated Warehouse Systems - Part II: Analysis and experimental results," *IEEE Trans. on Automation Science and Engineering*, vol. 9, no. 4, pp. 654–668, 2012.

[15] ——, "A cyber-physical view of automated warehouse systems," *2016 IEEE International Conference on Automation Science and Engineering (CASE'16), Fort Worth, TX, USA*, pp. 407–412, 2016.

[16] S. Buck, R. Hanten, K. Bohlmann, and A. Zell, "Generic 3d obstacle detection for agvs using time-of-flight cameras," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 4119–4124.

[17] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," in *Cooperative Robots and Sensor Networks 2015*. Springer, 2015, pp. 31–51.

[18] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, July 2006.

[19] L. Sabattini, V. Digani, M. Lucchi, C. Secchi, and C. Fantuzzi, "Mission assignment for multi-vehicle systems in industrial environments," *11th IFAC Symposium on Robot Control (SYROCO 2015), Salvador, Brazil*, vol. 48, no. 19, pp. 268 – 273, 2015.

[20] D. Morgan, G. P. Subramanian, S.-J. Chung, and F. Y. Hadaegh, "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1261–1285, 2016.

[21] P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson, "Wireless network design for control systems: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 2, pp. 978–1013, Secondquarter 2018.

[22] A. Burg, A. Chattopadhyay, and K. Lam, "Wireless communication and security issues for cyberphysical systems and the internet-of-things," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 38–60, Jan 2018.

[23] M. C. Lucas-Esta, J. L. Maestre, B. Coll-Perales, J. Gozalvez, and I. Lluvia, "An experimental evaluation of redundancy in industrial wireless communications," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, Sept 2018, pp. 1075–1078.

[24] B. P. Gerkey and M. J. Mataric, "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures," in *ICRA*, vol. 3, 2003, pp. 3862–3868.

[25] A. Yalaoui, H. Chehade, F. Yalaoui, and L. Amodeo, *Optimization of logistics*. John Wiley & Sons, 2012.

[26] F. Amato, F. Basile, C. Carbone, and P. Chiacchio, "An approach to control automated warehouse systems," *Control Engineering Practice*, vol. 13, no. 10, pp. 1223 – 1241, 2005.

[27] F. Basile, P. Chiacchio, and D. D. Grosso, "A control oriented model for manual-pick warehouses," *Control Engineering Practice*, vol. 20, no. 12, pp. 1426 – 1437, 2012.

[28] Y. Tatsumoto, M. Shiraishi, K. Cai, and Z. Lin, "Application of online supervisory control of discrete-event systems to multi-robot warehouse automation," *Control Engineering Practice*, vol. 81, pp. 97 – 104, 2018.

[29] V. Digani, M. A. Hsieh, L. Sabattini, and C. Secchi, "Coordination of multiple agvs: a quadratic optimization method," *Autonomous Robots*, vol. 43, no. 3, pp. 539–555, Mar 2019.

[30] M. Dotoli and M. P. Fanti, "A coloured petri net model for automated storage and retrieval systems serviced by rail-guided vehicles: A control perspective," *International Journal of Computer Integrated Manufacturing*, vol. 18, no. 2-3, pp. 122–136, 2005.

[31] M. P. Fanti, A. M. Mangini, G. Pedroncelli, and W. Ukovich, "A decentralized control strategy for the coordination of agv systems," *Control Engineering Practice*, vol. 70, pp. 86 – 97, 2018.

[32] I. Draganjac, D. Mikli, Z. Kovai, G. Vasiljevi, and S. Bogdan, "Decentralized control of multi-agv systems in autonomous warehousing applications," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 1433–1447, Oct 2016.

[33] K. Zheng, D. Tang, W. Gu, and M. Dai, "Distributed control of multi-agv system based on regional control model," *Production Engineering*, vol. 7, no. 4, pp. 433–441, 2013.

[34] L. Cancemi, A. Fagiolini, and L. Pallottino, "Distributed multilevel motion planning for autonomous vehicles in large scale industrial environments," in *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2013, pp. 1–8.

[35] V. Marik and D. McFarlane, "Industrial adoption of agent-based technologies," *IEEE Intelligent Systems*, vol. 20, no. 1, pp. 27–35, 2005.

[36] F. Basile, P. Chiacchio, and E. D. Marino, "Automated generation of a simulation model for the decentralized control of automated warehouse systems," *2018 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 540–546, 2018.

[37] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Hierarchical traffic control for partially decentralized coordination of multi agv systems in industrial environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6144–6149.

[38] ——, "Ensemble coordination approach in multi-agv systems applied to industrial warehouses," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 922–934, 2015.

[39] E. Semsar-Kazerooni and K. Khorasani, "Multi-agent team cooperation: A game theory approach," *Automatica*, vol. 45, no. 10, pp. 2205–2213, 2009.

[40] G. Cavone, M. Dotoli, N. Epicoco, D. Morelli, and C. Seatzu, "A game-theoretical design technique for multi-stage supply chains under uncertainty," *2018 IEEE International Conference on Automation Science and Engineering (CASE'18)*, pp. 528–533, 2018.

[41] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," in *Proceedings of the 19th National Conference on Innovative Applications of Artificial Intelligence - Volume 2*, ser. IAAI'07. AAAI Press, 2007, pp. 1752–1759.

[42] Z. Ma, D. S. Callaway, and I. A. Hiskens, "Decentralized charging control of large populations of plug-in electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 1, pp. 67–78, 2011.

[43] E. Roszkowska and S. A. Reveliotis, "On the liveness of guidepath-based, zone-controlled dynamically routed, closed traffic systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 7, pp. 1689–1695, Aug 2008.

[44] B. Horling and V. Lesser, "A survey of multi-agent organizational paradigms," *The Knowledge engineering review*, vol. 19, no. 4, pp. 281–316, 2004.

[45] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. C-29, no. 12, pp. 1104–1113, Dec 1980.

[46] J. Peng and S. Akella, "Coordinating multiple double integrator robots on a roadmap: Convexity and global optimality," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA'05), Barcelona, Spain*, pp. 2751–2758, April 2005.

[47] J. P. Cavada, C. E. Corts, and P. A. Rey, "A simulation approach to modelling baggage handling systems at an international airport," *Simulation Modelling Practice and Theory*, vol. 75, pp. 146 – 164, 2017.

[48] H. Hu, Y. Liu, and M. Zhou, "Maximally permissive distributed control of large scale automated manufacturing systems modeled with petri nets," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 2026–2034, Sept 2015.

[49] F. Yang, N. Wu, Y. Qiao, and R. Su, "Polynomial approach to optimal one-wafer cyclic scheduling of treelike hybrid multi-cluster tools via petri nets," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 270–280, Jan 2018.

[50] F. Basile, P. Chiacchio, and D. Del Grosso, "A two-stage modelling architecture for distributed control of real-time industrial systems: Application of UML and Petri net," *Computer Standards & Interfaces*, vol. 31, no. 3, pp. 528–538, March 2009.

[51] F. Basile, P. Chiacchio, and D. Teta, "A hybrid model for real time simulation of urban traffic," *Control Engineering Practice*, vol. 20, no. 2, pp. 123 – 137, 2012.

[52] F. Basile, P. Chiacchio, A. Giua, and C. Seatzu, "Deadlock recovery of Petri net models controlled using observers," *8th International Conference on Emerging Technologies and Factory Au-*

tomation, (ETFA'01), Antibes, France, vol. 2, pp. 441–449, Oct 2001.

[53] D. You, S. Wang, and M. Zhou, "Synthesis of monitor-based liveness-enforcing supervisors for s^3pr with ξ-resources," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 6, pp. 967–975, 2015.

[54] H. Hu, M. Zhou, and Z. Li, "Supervisor optimization for deadlock resolution in automated manufacturing systems with petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 4, pp. 794–804, Oct 2011.

[55] Automatic Control Group of University of Salerno, "Pnetlab." [Online]. Available: http://www.automatica.unisa.it/pnetlab.php

[56] F. Basile, C. Carbone, P. Chiacchio, R. Boel, and C. Avram, "A hybrid model for urban traffic control," *2004 IEEE International Conference on Systems, Man and Cybernetics (SMC'04)*, vol. 2, pp. 1795–1800, 2004.