UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Dipartimento di Matematica Renato Caccioppoli
Dottorato di Ricerca in Matematica e Applicazioni

Tesi di Dottorato in Matematica e Applicazioni

Vehicular ad Hoc Networks: an algorithmic and a game-theoretic approach

Anno Accademico 2021/2022

Relatore
Ch.mo prof. Aniello Murano
Ch.mo prof. Walter Balzano

Candidato
Silvia Stranieri

To my sisters, so that you can always fight for your dreams.

Abstract

With the fast growth of population in the city, the traffic congestion as, well as the parking problem, becomes over and over a problem to deal with, since they reduce road safety and increase stress-level for drivers. About this, Vehicular ad hoc Networks provide a useful framework to handle traffic-related problems, allowing a wireless communication among vehicles. In this work, the traffic congestion problem and, in particular, the smart parking, are addressed from two different points of view: an algorithmic one, providing a step-by-step procedure that answers to a given question, and a game-theoretic one, by modeling the problem as a multi-agent game, whose solution is formally defined as a winning strategy. Regarding the algorithmic approach, first, an innovative signal-based representation of Vehicular ad hoc Networks is provided. This kind of representation is modeled in a way to highlight crowded areas of the network, assigning a value (the *congestion factor*) to each vehicle according to its position in the network. The representation is made canonical so to make comparison operations easy and intuitive. Then, a smart parking algorithm ex-

ploiting the blockchain mechanism is proposed. The setting scenario involves a consortium of car parks to complete the parking process, and the blockchain consensus rule allows guaranteeing fair competition among members. Concerning the game-theoretic approach, instead, first reachability results on both turn-based and concurrent Dynamic Epistemic Logic games are proved, then the parking problem is modeled as a multi-agent game, in which drivers looking for an available parking space are players trying to win a game. The solution is found through a social equilibrium, namely the Nash equilibrium, that turns out to be a good compromise between feasibility and optimality.

Contents

Abstract **ii**

I Introduction **1**

II Part One **8**

1 A Signal-Based Model for Vehicular ad Hoc Networks **9**

1.1 Introduction 9

1.2 Related Work 15

1.3 Preliminaries 16

1.3.1 Dynamic Programming for Similarity Measure 16

1.4 Construction Phase 18

1.5 Signal Rendering and Examples 21

1.6 Signal Sampling and Feature Extraction 30

1.6.1 DTW Limits 30

1.6.2 Sampling 32

1.6.3 Feature Extraction 34

1.7 Local Alignment: Motivations 35

1.8 Similarity Computing 36

1.9 Evaluation 39

2 Algorithms for Smart Parking 45

2.1 Introduction 45

2.2 Related Work 48

2.3 Ant-Colony-Optimization-Based Parking Algorithm . . 50

2.3.1 Preliminaries 51

2.3.2 Towards ACO for Parking 58

2.3.3 Model Definition 58

2.3.4 Graph Coloring 59

2.3.5 The Pheromone Model 62

2.3.6 Constraint Relaxation 63

2.3.7 The ACO Algorithm for Parking 64

2.3.8 Valuation 66

2.3.9 Simulation Results 66

2.3.10 Benchmarks 67

2.4 Competitive-Blockchain-Based-Parking System with Fairness Constraints 69

2.4.1 Background 70

2.4.2 BC Parking Algorithm 77

2.4.3 The Consensus Algorithm 84

2.4.4 Model Validation 88

2.4.5 Benefits 91

III Part Two 93

3 Reasoning About DEL games 94

3.1 Introduction 94
3.2 Related Work 99
3.3 Background on Games and Epistemic Planning 101
3.3.1 Notations 101
3.3.2 Game arenas 101
3.3.3 The classic DEL setting 107
3.3.4 Generated structure 112
3.3.5 Epistemic planning 113
3.4 Controller Synthesis 115
3.4.1 The controller synthesis problem 115
3.4.2 The case of non-expanding action models . . . 119
3.4.3 The case of propositional action models 130
3.5 Distributed Strategy Synthesis 134
3.5.1 Setting up the game 135
3.5.2 The distributed strategy synthesis problem . . 137
3.5.3 Undecidability for two existential players . . . 139
3.5.4 The case of non-expanding action models . . . 143
3.5.5 Propositional actions+hierarchical information 147
3.6 Concurrent Games 149
3.6.1 Concurrent Actions 151
3.6.2 Conflicts 152
3.6.3 The role of the scheduler 154
3.6.4 Concurrent update product 155
3.6.5 Concurrent Games 157
3.6.6 Decidability for public actions 162

3.6.7 Decidability for propositional actions + hierarchical information 164

4 A Comparative Study on the Most Common Model Checking Tool: MCMAS 168

4.1 Introduction 168
4.2 Background on SL and MCMAS 170
4.2.1 Strategy Logic with Simple Goals 170
4.2.2 MCMAS 171
4.3 Comparison of Existing Tools 172

IV Part Three 177

5 A Nash-Equilibrium-Based Parking Algorithm 178

5.1 Introduction 178
5.2 Related Work 180
5.3 A Real Scenario 182
5.4 Parking Game Structure 184
5.5 The Parking Slot Selection Game 188
5.5.1 A Running Example 189
5.5.2 A Greedy Solution 189
5.5.3 Nash Equilibrium Based Solution 190
5.5.4 A Solution to the 3-players-3-slots Game 192
5.5.5 A Solution for the PSSG 193
5.6 Evaluation 197
5.7 Benchmarks 198

V Conclusions **201**

6 Conclusion **202**

Part I

Introduction

Vehicular ad hoc networks [7] (VANETs, for short) were my first approach to the research world during my bachelor thesis. This kind of network caught my attention immediately because they seemed to be very promising in the Internet of Things [14] and Artificial Intelligence [97] fields. At that time, I started studying VANETs, how they work, how the communication among vehicles is performed, what are the main challenging and critical aspects, but also their possible applications. When I started doing research on the topic at the beginning of my PhD, the first approach was algorithmic, trying to apply what I had learned during my university studies. First, a problem is clearly defined, then a procedure that provides an answer to the problem is built. The instructions of the defined procedure, executed step by step, lead to a correct solution to the problem. I could say that the first half of my PhD reflects exactly this kind of approach; I studied and provided several algorithmic solutions to open problems about VANETs [6]. These are wireless sensor networks in which vehicles play the role of sensors, exchanging information via *broadcasting* communication. It is pretty intuitive how strong is the potential of such a framework, to handle traffic management, to guarantee road safety, but also to improve the quality of viability and to reduce air pollution. During my PhD, I investigated several challenging related topics, some of which are explained in detail in this work, also with a particular focus on smart parking applications. Some of them are

summarized in the following:

- A compact and less expensive representations of vehicular networks: typically, VANETs are represented as directed graphs where vehicles are nodes and their communication links are edges. What has been proposed, instead, is signal-based representation, by exploiting an easy two-dimensional space mapping of the network, highlighting the traffic distribution over the network [22, 21].

- A direction-based clustering algorithm for VANETs: indeed, very often clustering in VANETs is performed according to vehicles punctual position, that in such a dynamic environment can lead to mistakes. The proposal is based on a preliminary study of the direction of movement of each vehicle, expressed in terms of angle of displacement with respect to a fixed point, whose position is known [72].

- A behavioral clustering for VANETs, providing a framework that processes information about vehicle behaviors aiming at extract a similarity measure to establish if two vehicles should belong to the same cluster [23].

- An algorithm for smart allocation of vehicles into parking slots by exploiting the behavior of real ants when they look for food for the path selection [25, 2].

- A destination-based parking algorithm, that allows allocating vehicles in parking slots exploiting existing algorithm for memory allocation in operating systems (Best, First, and Worst fit), by also considering the destination that the driver wants to reach [19].

- A blockchain-based parking algorithm that simulates the parking process in a consortium of car parks, where the blockchain mechanism is employed to guarantee a fair competition among members [20].

At the end of my first PhD year, I started studying game theory and strategic reasoning [100]. This world led me to a much more theoretical approach to problems, precisely a *game-theoretic* approach, but as interesting and effective as the algorithmic one. It concerns formal models to define the interaction among agents. When the model is formally defined, the *model checking*, a formal technique to validate models, can be exploited to prove if some desired properties hold in the model. I have been attracted to this kind of know-how, and it was clear that such an approach could have brought several advantages. Indeed, formal verification allows finding possible errors or incorrect behaviors of a system. In some systems, failures can also cause the loss of human life, think of a plane malfunction, but also loss of money and time, in the most typical scenario. For this reason, in the second part of my PhD, I studied the logics and the formalisms behind the

definition of desired properties in a model, by getting closer to the multi-agent reachability games with imperfect information world. Indeed, these games have several applications, from economics, to video games and robotics, but they are also undecidable in several cases. For this reason, possible assumptions on the knowledge of the players involved have been studied, such as *hierarchical information*, or on the kind of actions, such as *public actions* and *public announcements*, so to make the strategy synthesis decidable. To do so, *Dynamic epistemic logic* [129] has been considered to model high-order knowledge (agent A knows that agent B knows etc) and how player's actions affect the world and the way it is perceived by themselves.

During the last year, I had the chance to go through multi-agent systems more deeply because I had the pleasure to collaborate with Imperial College University on a comparative study on the existent versions of one of the most employed model-checkers for multi-agent systems, MCMAS [80], to evaluate possible further extensions.

At the end of my path, by following the lead of the previous works, my studies moved toward a way to apply the formal verification techniques to the vehicular field which I was familiar with, in particular to the smart parking application, by formally defining the parking process as a game, and a corresponding model to express it.

I could proudly say that this thesis is the result of the two main influences and mentors I had during my studies, and it is written by fol-

lowing the experiences made during the PhD. Indeed, it is a crescendo of awareness of the available techniques and technologies to solve everyday problems in vehicular context, such as the smart parking.

Given the different flavors of the themes treated in this work, for better readability, I decided to maintain the same evolution of my research studies even in the thesis structure. For this reason, the work is organized in *parts*, one for each phase of my PhD studies: (*i*) the first part is about an algorithmic approach to VANETs, and in particular to the smart parking problem; (*ii*) the second part, instead, regards a high-level game-theoretic approach on reachability games, with a particular study on MCMAS, a know model checker for multi-agent systems; (*iii*) the third part, finally, is dedicated to the application of the game-theoretic approach to a case of study in VANETs environment, namely the smart parking problem in Federico II Hospital Company. Each part, furthermore, is divided in *chapters*, each of whom goes through a specific research topic: (*i*) chapter one provides a signal-based representation of a VANET, in contrast with the standard graph-based one typically used, highlighting the benefits coming from such a representation, in particular for comparison among networks; (*ii*) chapter two provides two algorithmic solutions to the smart parking problem, first by exploiting the Ant Colony Optimization mechanism, then by introducing the blockchain features that add a fairness property; (*iii*) chapter three is about distributed

synthesis on reachability games, and their restriction to subsets of actions; (*iv*) chapter four goes through the most common model checker for multi-agent systems (MCMAS) by providing a comparative study on the existing current versions and how they work on fragments of Strategy Logic, an emerging logic for strategic reasoning; finally, (*v*) chapter five constitutes the meeting point of the previous chapter, where a very concrete problem, the smart parking one, is solved as a multi-agent game through a known social equilibrium, the Nash Equilibrium. Each chapter is *stand-alone*, with its own introduction and related work sections. The work is completed with a technical *conclusions* chapter, where the results obtained are specified with more detail.

Part II

Part One

Chapter 1

A Signal-Based Model for Vehicular ad Hoc Networks

1.1 Introduction

Vehicular ad hoc networks are a recent research topic that is keeping providing new challenges and input for further innovation aspects. The main goal is to guarantee a continue communication among vehicles, by not relying on any central controller [125]. The information exchange happens via *broadcasting*, meaning that when a vehicle releases data, it will be caught by any other vehicle close enough. The concept of *close enough* is determined by the Received Signal Strength

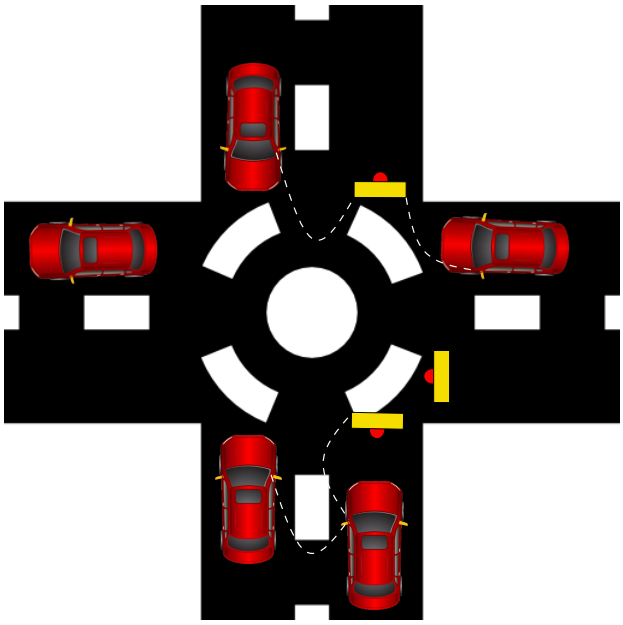


Figure 1.1: VANETs configuration

Indicator (RSSI), that measures the power received in a radio signal [134]: the greater the RSSI the stronger the signal. The RSSI values are measured in dBm and have typical negative values ranging between 0 dBm (excellent signal) and -110 dBm (extremely poor signal)[103].

Vehicles in the network are all equipped with onboard side units so they can communicate directly among them if they are in the communication range of each other (meaning that the RSSI is powerful enough to catch the signal). In this case, we speak about a vehicle-to-vehicle communication (V2V). In order to increase the communication power of VANETs, also roadside units are placed along the roads so that infrastructure items (traffic lights, for instance) can act as a bridge to make two far vehicles exchange information.

Figure 1.1 represents a typical VANET scenario, where vehicles are approaching a roundabout. Some vehicles are close enough to perform

a V2V communication, some others, instead, use traffic lights as a bridge.

VANETs application fields are several and we can summarize them as follows [62, 64]:

- Safety: according to what said by World Health Organization (WHO, for short), approximately 1.3 million people die each year as a result of road traffic crashes (information updated on 21 June 2021). Also, it emerges the dramatic fact that road injuries are the cause of death for children and young adults aged 5-29 years. VANETs can help reduce those crashes by alerting drivers in time, by managing collision avoidance, traffic sign notification, incident management.

- Efficiency: TomTom performed a study on traffic congestion in 2020, whose graphic result is in Figure 1.2. They covered 416 cities across 57 countries on 6 continents [124], highlighting a congestion level of 54% in Moscow, 53% in Manila, and 51% in Kyiv, to name a few. VANETs self-organizing capability can be exploited to handle such a congestion level, by applying traffic management and traffic monitoring techniques.

- Quality: VANETs intrinsic nature lends itself to provide different facilities for users, such as electronic payments, parking information, and so on. They can be used both for entertainment applications and background information ones.

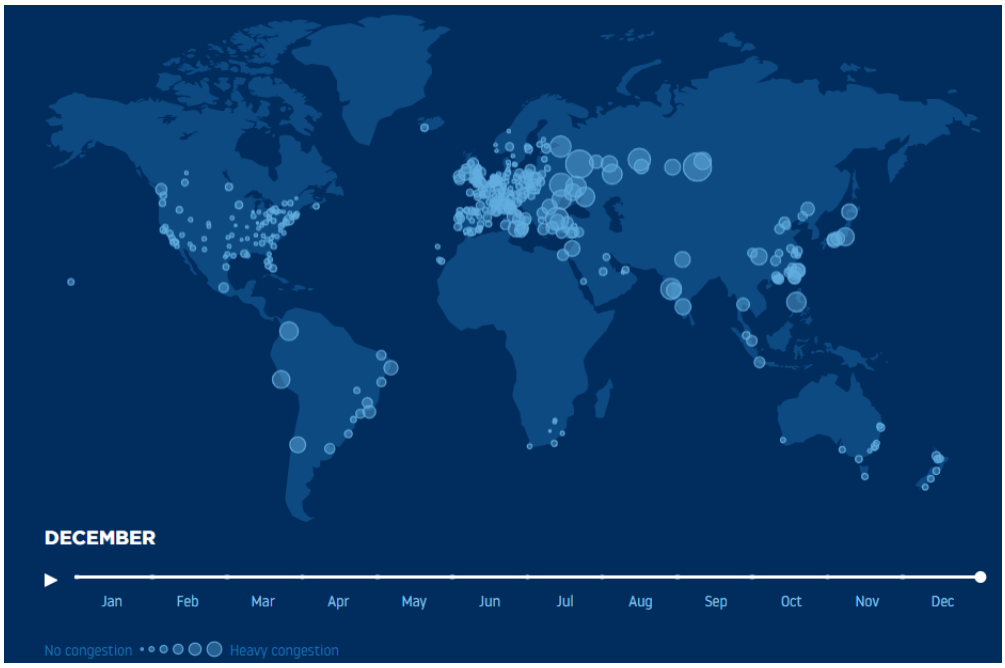


Figure 1.2: Traffic Index Chart by TomTom [124]

Vehicular ad hoc networks raise several challenges, that are summarized as follows [62]:

- Node Velocity: in VANETs very different velocity have to be handled, from inactive RSUs to vehicles moving at 100 km per hour. This structural obstacle is very often managed through clustering algorithms, so to group vehicles with same characteristics.

- Movement Patterns: the majority of movements of the nodes in a VANET follows predefined directions. Precisely, the trajectory of movement is quite predictable, except for roundabouts, or road crossings.

- Node Density: the number of vehicles in the communication

range. Many researchers assign a trustworthiness to vehicles.

- Security and Privacy: the exchange of messages among vehicles should be safe and should protect the privacy of drivers.

In the typical representation, a traffic network is seen as a non-directed graph $G = (V, E)$ in which the set of nodes V corresponds to the vehicles in the network, with $|V| = n$, and the set of edges E corresponds to the links, if any, between the nodes, with $|E| = m$. Let us define R as the *distance coefficient*, i.e. the maximum value within which two vehicles can communicate with each other. Such a value is calculated referring to the measurement of the power present in a received radio signal, as defined before. In particular, it is intuitive to say that increasing the distance, the RSSI decreases, by definition.

Moreover, given two nodes $\mathtt{u}, \mathtt{v} \in \mathtt{V}$, let $d(u, v) \in [0, R]$ be the euclidean distance between u and v. We can state that an edge $(u, v) \in E$ **exists** iff $d(u, v) \leq R$, and $weight(u, v)$ is its weight.

Since in a vehicular network $d(i, j) \neq d(j, i)$ (with $i \neq j$), let us say that a normalized graph is obtained by computing the average of these two distances and by assigning this value to the edge linking i and j, so that the corresponding adjacency matrix is symmetric. Finally, for each $v \in V$, $neig(v) = \{v' | (v, v') \in E\}$ and $degree(v) = |neig(v)|$ are the set of neighbors of v and its size, respectively.

Congestion is a serious problem affecting roads of all the continents as previously seen in Figure 1.2, and it is a challenging issue in

VANETs, as pointed out in [37]. For this reason, traffic awareness can help drivers making the right decision about the path to choose to get to a given destination. Moreover, considering the high dynamism of vehicular ad hoc networks, the necessity of real-time responses raises the need for a network representation that is faster to handle and more intuitive, with respect to the standard graph one, and that, not only, catches as much information as possible, but also allows dealing with congestion.

In the sequel, a new signal-based technique to represent a vehicular network is provided. Such a representation is built by mapping a graph onto a two-dimension space, where the first dimension is the vehicles in the network, and the second one is obtained by computing an opportune *congestion factor* for each vehicle, expressing how much that vehicle is in a crowded situation. The signal-based version ends up being more intuitive and meaningful with respect to the standard one and, by providing an opportune comparison model, it also allows performing interesting computations over pairs of networks, so to extract possible similarities. To do so, a similarity measure is provided as a probability value between 0 and 1, by performing an analysis of the signal, aimed at the extraction of the most significant parameters that characterize it. Precisely, it is first performed a sampling phase, where the sampling rate is chosen, then a feature extraction phase, needed to obtain the signal characteristics. The final similarity is computed first

through an area-to-area comparison, then by applying local alignment techniques.

1.2 Related Work

Vehicular ad hoc networks research field, placed into the bigger one of Intelligent Transportation Systems [140], is very active nowadays. The communication system promoted has been proven to be very efficient for road safety, but also useful for traffic management. One of the most challenging aspects in vehicular context is the overcrowding management. Indeed, the high-density nature of VANETs may impact the efficiency of the system. In this work, for the first time, the overcrowding problem is addressed through a smart representation of vehicular network [22], enriched with a comparison model to recognize similar behaviors [21].

Let us first analyze the state of the art on representation matter for VANETs and comparison models. In literature, the most popular representation for vehicular ad hoc network is graph-based, such as in [67], where authors use a graph representation based on HashMap to reduce the computation time. It is necessary to mention [114], where the authors, in order to face the multi-class classification problem, implement the traffic sign recognition aimed to the individuation of similar behaviors, proposing two techniques based on *machine learning* approach and fuzzy regression trees respectively. Authors of [79],

instead, step away from VANETs by focusing on song similarity, classifying them according to their content. The key point is in the spectrum of the file, whose information are used in combination with the known k-means algorithm, with the aim of grouping similar samples of the signal. Finally, in [101], they introduce a novel concept of trainable similarity measure, relying on images. Each image represents a traffic situation, and it is divided in regions, such that similar components can be studied region-wise.

1.3 Preliminaries

As follows, I'm going to present a used technique for string alignment that will be used to specify the proposed comparison model.

1.3.1 Dynamic Programming for Similarity Measure

Very often, methods for sequence alignment are based on dynamic programming algorithms. The sequence alignment is a very known biological problem, whose aim is to assign a score expressing how similar the two sequences are [54], and it consists of transforming a sequence into another through edit operations (insertion, deletion, and substitution) [1].

A dynamic programming algorithm has a bottom-up phase to fill

a score matrix, and a trace-back phase to recover the best alignment, as pointed out in [1, 54]. It is commonly known that two classical dynamic-programming-based alignment algorithms exist, and they are compared in [1]:

- Global alignment, aimed at finding the best match of the entire sequences;
- Local alignment, aimed at finding similar region of different sequences.

Precisely, to perform a signal representation the focus is on the second form of sequence alignment, that also seems to be more suitable to perform comparisons, as will be explained later. In such an algorithm, given m and n the lengths of the sequences to be aligned, the score matrix M has size $m + 1 \times n + 1$. The score is computed according to some parameters:

- g is the gap penalty score;
- mi is the mismatch score;
- ma is the match score.

The way it is filled is explained in the following [1].

Definition 1.3.1 (Score Matrix Initialization)**.**

$$M[i, 1] = M[1, j] = 0 \quad \forall i \in [1, m], j \in [1, n]$$

Definition 1.3.2 (Bottom-up Phase)**.** *Let us define a function compare assigning the score of match or mismatch according to the current symbol of the two sequences s_1 and s_2:*

$$compare(i,j) = \begin{cases} ma & if\ s_1[i] = s_2[j] \\ mi & otherwise \end{cases}$$

The score matrix is filled as follows:

$$M[i,j] = max \begin{cases} M[i-1,j] & + & g \\ M[i,j-1] & + & g \\ M[i-1,j-1] & + & compare(i-1,j-1) \\ 0 & & \end{cases}$$

Definition 1.3.3 (Trace-back Phase)**.** *The alignment recovery starts with the maximum entry of the score matrix and traverses it diagonally until the first 0-entry is met.*

1.4 Construction Phase

The first step to realized the model as explained before is to build a smart vision of vehicular networks based on signal representation, starting from randomly generated points in the space (based on the normal distribution). In order to obtain a signal that reflects the congestion of any vehicle in the network regardless the way it is visited,

we compute a *congestion factor* through a function f for each node of the network:

$$f : V \longmapsto [0, 1] \tag{1.1}$$

This factor is parametric on R and is computed as the difference between the *ideal congestion* and the *local congestion*. Given the node v, the ideal congestion represents the situation where all the neighbors are at maximum distance from v:

$$ideal(v) = degree(v) * R \tag{1.2}$$

Instead, the local congestion is an arithmetic average over the adjacents of v:

$$local(v) = \frac{\sum_{u \in neig(v)} weight(u, v)}{degree(v)} \tag{1.3}$$

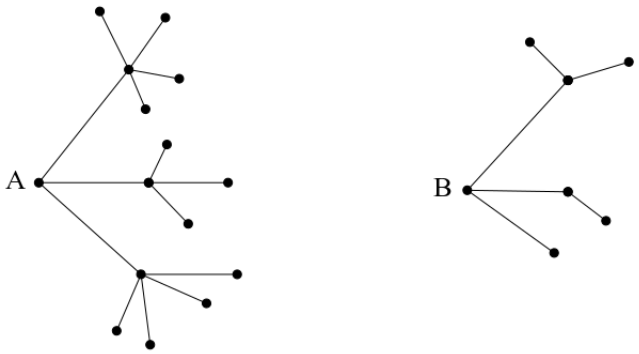


Figure 1.3: Example of misleading congestion factor computation

One can observe that the congestion degree of a node does not depend only on the number of its neighbors, but mostly on their congestion. Thus, we need to take into account the congestion of each

neighbor of the considered node, in order to avoid the situation as the one shown in Figure 1.3, in which the nodes A and B have the same number of neighbors, but the congestion of such neighbors is different. Indeed, in the left side we have a higher congestion level with respect to the right side, but with the previous formula A and B would have had the same congestion factor.

For this reason, we introduce a *weighted ideal congestion*:

$$weighted\_ideal(v) = \sum_{u \in neig(v)} (degree(u) - 1) * ideal(v) \qquad (1.4)$$

Notice that when we compute the degree of the neighbors of a given node v, we decrease it by 1 in order not to consider v again. Once the congestion factors are computed, they are normalized, dividing them by the maximum congestion factor of the considered network.

The final network representation is $f(V)$, with f as defined in equation 1.1. A key point in the wave representation is the order in which the nodes are placed on the x axis. The goal is to obtain a signal that immediately highlights the most congested areas and which points belong to them. For this reason, it has been chosen a cluster-oriented visit of the network that, starting from a random point, continues the visit of neighbors, putting them in the same connected area, as long as a certain distance is not overcome. With "connected area" we mean a set of related nodes. It is important to notice that this is not a new clusterization technique, but an alternative way of visiting a network

by detecting areas of nodes connected within a certain distance. To this aim, we introduce a tolerance ϵ such that, given a node v and its neighbor u, if $d(v,u)) \leq R-\epsilon$, u is in the same connected area as v. It is also worthy to point out that the congestion factor of isolated nodes are distinguished by the one of nodes having only edges with weight $\geq R-\epsilon$, by preserving them in the network. Once the connected areas are obtained, the nodes are placed on the x axis in such a way that the ones belonging to the same connected area are contiguous. In particular, for each connected area, the elements belonging to it are placed in increasing order with respect to the congestion factor. This choice makes easier the detection of connected areas just looking at the signal.

1.5 Signal Rendering and Examples

In this section, some examples of network transformation are provided, by also analyzing all the information that can be inferred. The examples are generated according to the following parameters:

- the number of nodes n,
- maximum distance R allowed between two nodes,
- a tolerance ϵ introduced in the section above,
- the standard deviation sd, used to produce random points through a normal distribution.

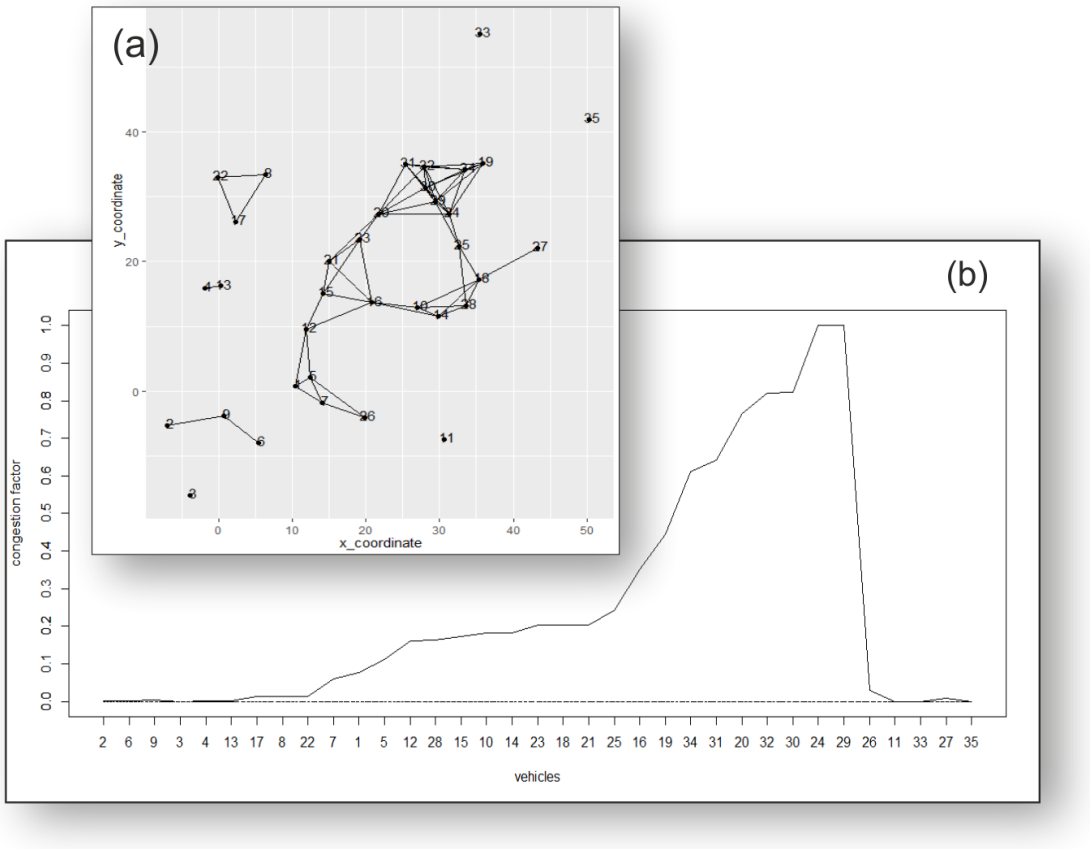


Figure 1.4: Random generation of a network with $n = 35$, $sd = 10$, $R = 10$, $\epsilon = 0.3$ (a) and the corresponding wave representation (b)

Starting from the signal representation, it is easy to understand the network congestion, as well as identify the different connected areas. Indeed, according to how the signal is built, a null congestion factor corresponds to an isolated node, and each high-low transition induces a new connected area, but the opposite does not hold in general. Hence, this is a *necessary* but not a *sufficient* condition for the starting of a new area. This means that there could be changes of area hidden by the signal, when the highest congestion factor of the first connected area is smaller than the lowest one of the second connected

area, introducing *false negatives*. As shown in Figure 1.4(b), which is the signal corresponding to the graph 1.4(a), it is intuitive to observe that the most congested nodes are 24 and 29. Another information easy to deduce is about the areas of the network made by single nodes. Indeed, they are identified by the points whose congestion factor coincides with the dashed line, i. e. 3, 11, 33, and 35. In order to detect the remaining connected areas, we need to retrieve the high-low transitions, corresponding in this network to the points 9, 29, 26, and 27. By analyzing the network in Figure 1.4(a), we would have expected a change of area between the nodes $4, 13$ and $8, 17, 22$, that is hidden by the signal because of false negatives. The same happens between the nodes 22 and 7.

The signal obtained as shown in the example, is determined by visiting the nodes of the network starting from the one having the smallest x-coordinate. By changing the starting point, clearly the congestion factor of each node stays the same, but the resulting wave can be a permutation of the peaks in the current signal. This could be a limitation for comparisons between signals. For this reason, a *canonical form* of the signal is introduces, obtained by changing the order of the nodes on the x axis: not only the nodes are ordered in increasing order of congestion factor inside each connected area, but also each connected area is ordered in decreasing order of maximum congestion factor on the x axis. Through this normalization, we obtain a signal

having the connected area with the highest congested node on the left side, and the single nodes on the right side.

With this approach, false negatives are also reduced. Indeed, it is no longer possible that the highest congested factor of a previous connected area is smaller than the lowest factor of the next area, since they are ordered, but false negatives can still occur, as formally reported in the following theorem:

Theorem 1.5.1. *Given two successive connected areas a_1 and a_2, let h_i and s_i (with $i \in [1, 2]$) be the highest and lowest congestion factors respectively for the corresponding areas, then (**premise**) if a false negative occurs, it means that (**conclusion**) a_2 is made of a single node having h_1 as congestion factor.*

Proof. Let us assume the premise true, thus we have a false negative. The following inequalities hold:

$$\begin{cases} h_1 \geq h_2 & \text{by construction} \\ h_1 \leq s_2 & \text{by definition of false negative} \\ s_2 \leq h_2 & \text{trivially} \end{cases} \tag{1.5}$$

Hence, the only possibility is that $s_2 = h_2 = h_1$. □

The settings of the first example in Figure 1.5 generates a fully connected network, with a high amount of nodes in overcrowding. Indeed, the corresponding signal reflects these features, since we do not

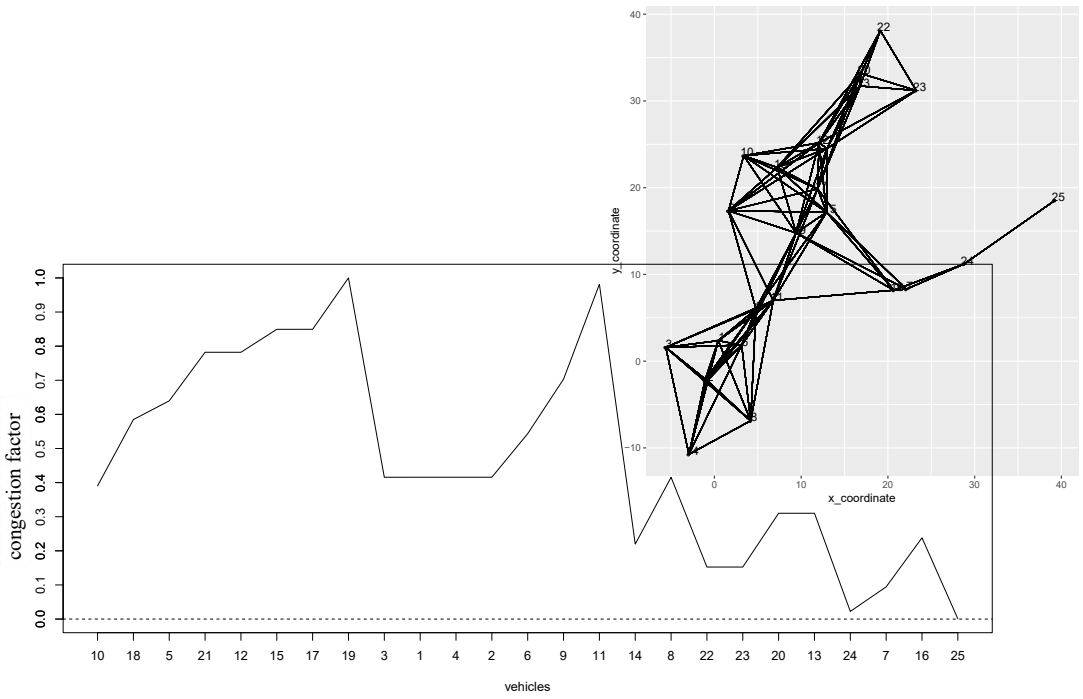


Figure 1.5: Above, a network generated at random with 25 nodes, $r = 15$, $\epsilon = 0.5$, and standard deviation 10. Below, the corresponding signal.

see nodes with zero congestion factor, while we see, instead, the majority of nodes belonging to areas with high congestion factors.

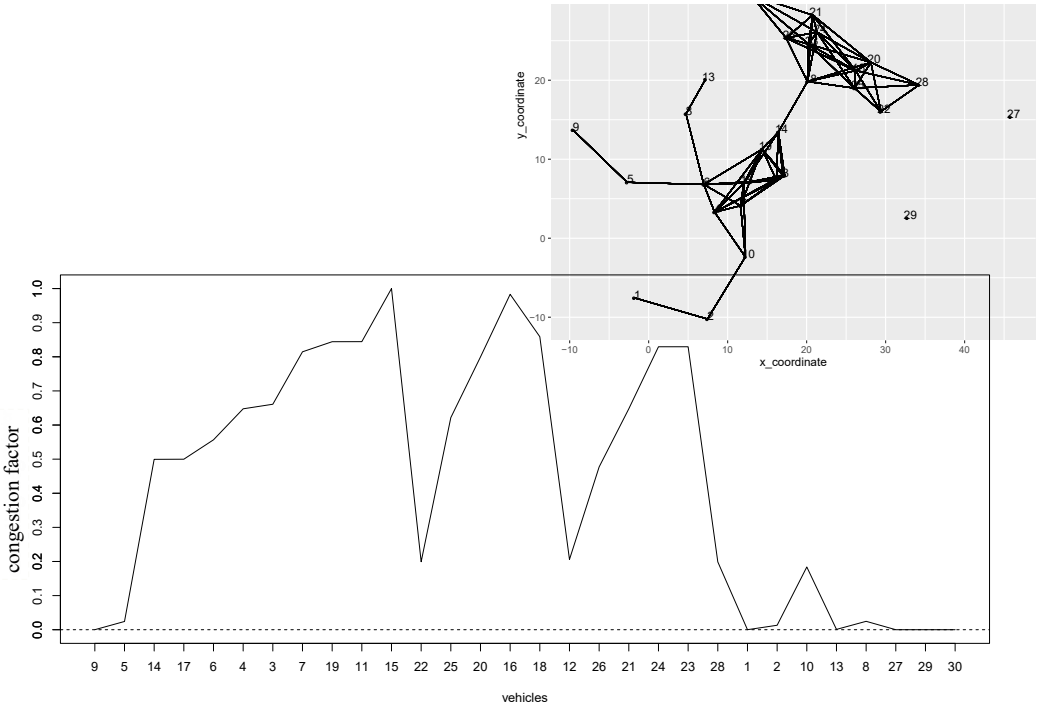


Figure 1.6: Above, a network generated at random with 30 nodes, $r = 10$, $\epsilon = 0.2$, and standard deviation 10. Below, the corresponding signal.

Changing the tolerance as in the second example of Figure 1.6 leads to a less connected network, populated also with isolated vehicles, corresponding to the zero congestion factor nodes on the signal (precisely, 27, 29, and 30), with a huge agglomeration of vehicles, corresponding to the first area of the signal, and other smaller groups of vehicle.

The standard deviation increasing of the third example of Figure 1.7, instead, produces a very connected network, with a single high overcrowding vehicle, reflecting in the node with congestion factor 1 in the corresponding signal.

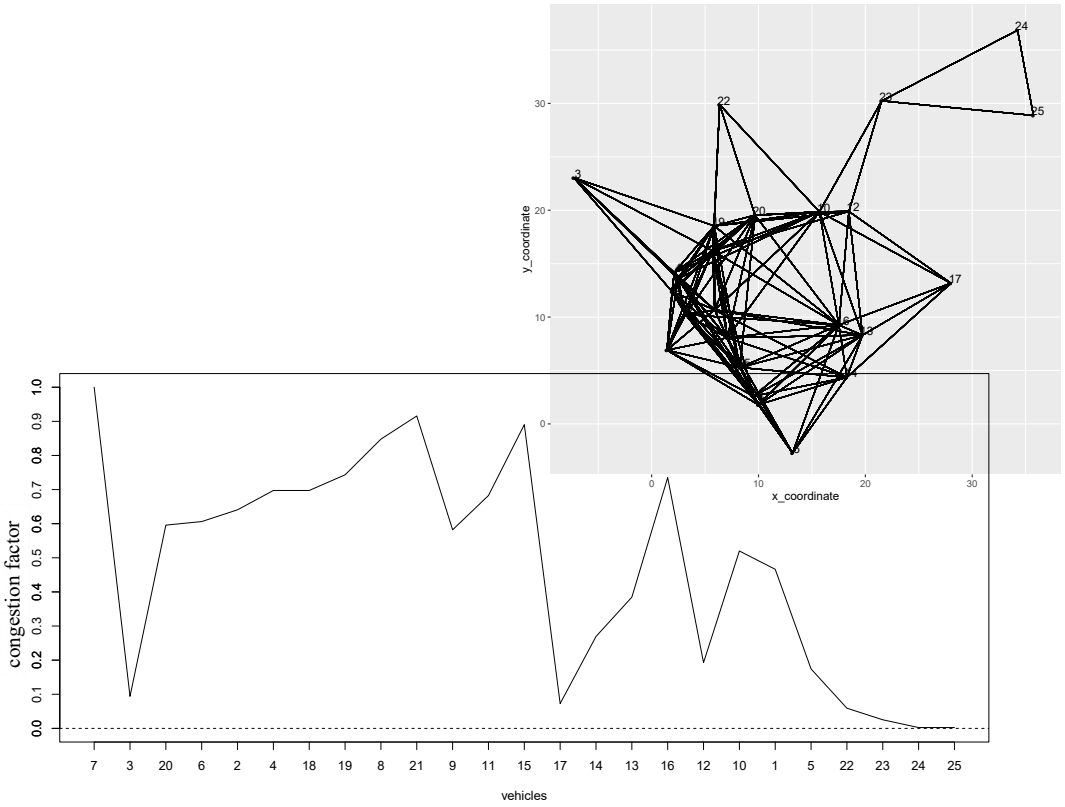


Figure 1.7: Above, a network generated at random with 25 nodes, $r = 15$, $\epsilon = 0.5$, and standard deviation 8. Below, the corresponding signal.

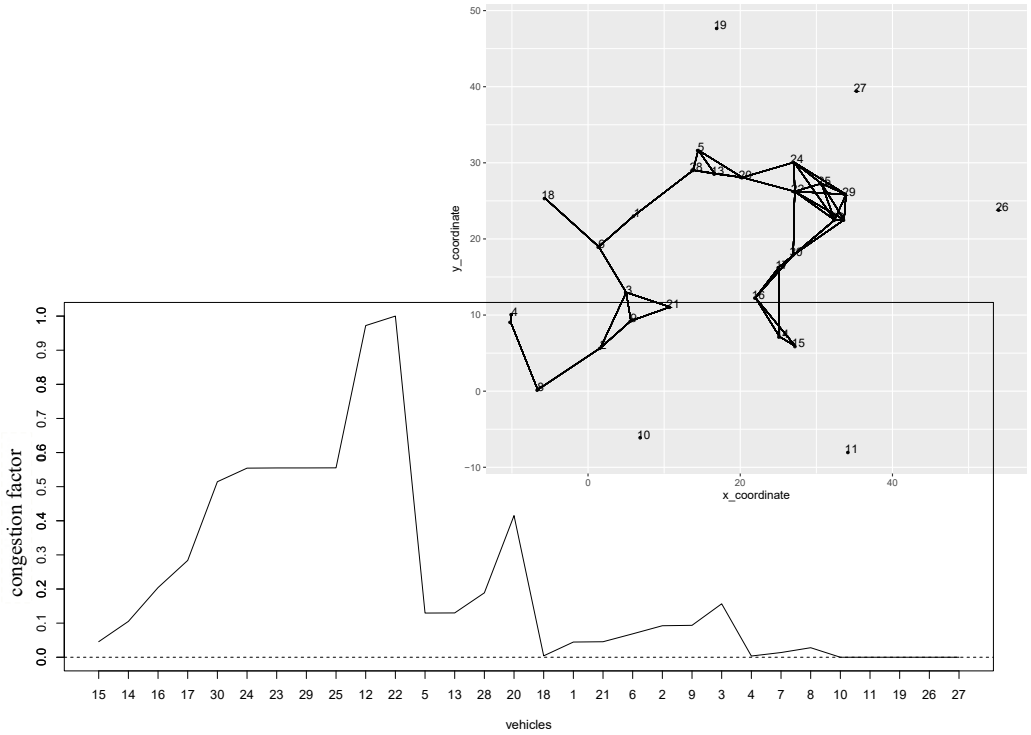


Figure 1.8: Above, a network generated at random with 30 nodes, $r = 10$, $\epsilon = 0.2$, and standard deviation 12. Below, the corresponding signal.

Finally, the fourth example in Figure 1.8 has been generated by decreasing both the tolerance and the standard deviation. The result is a very disconnected network with few overcrowding vehicle, leading in the first area of the signal, and many vehicles with low, or even zero, congestion factor.

1.6 Signal Sampling and Feature Extraction

After building network configuration as a two-dimensional signal, our concern now is how to compare these signals in such a way to have meaningful information. According to how the signal has been built, the most powerful information is given by the maximum crowding degree of each neighborhood which, in terms of signal, is represented by the peaks. For this reason, the idea is to use the dynamic programming approach to find the best alignment between two sequences which, in this special case, are the signal representation of two networks. Such a technique leads us to a measure expressing how much the considered sequences are similar. One of the most known methods applying this approach is the Dynamic Time Warping, also known as DTW.

1.6.1 DTW Limits

Dynamic Time Warping [96] is a known method to find the optimal alignment between two sequences. Typically, DTW is applied to time-dependent sequences, since it is based on temporal axis stretching aimed to align similar parts of the them. DTW is implemented by means of dynamic programming, and global alignment in particular. Using DTW-oriented approach to extract a similarity measure between signals, in our singular case, might not be the best option. Indeed, in

our representation of the signal, the x axis is characterized by the nodes of the network (and not the time). For this reason, a stretching of the x axis produces a not likely alignment, since the network configuration is distorted.

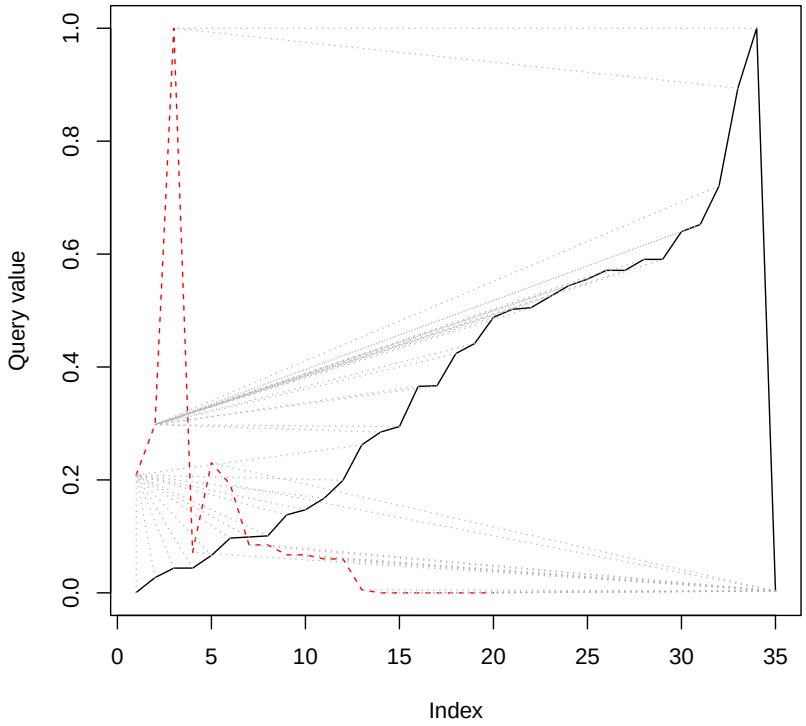


Figure 1.9: DTW alignment between two signals (the first one in dotted line and the second one in solid line) representing two different traffic situations.

As we will explain later, experiments show that network representation comparisons through DTW very likely produces a high similarity value, even if the two starting signals seem very different. This is due to the temporal stretching that always allows to find a good alignment between them.

An example is shown in Figure 1.9, representing how the DTW algorithm aligns two signals, that we assume having been generated starting from two different networks, as explained in [22]. Intuitively, they correspond to very different traffic situations: indeed, the dotted signal represents a configuration in which we can identify an area made of few nodes that are highly congested, some other areas made of nodes with a medium-low congestion value, and different isolated nodes that are not congested at all; the solid signal, instead, presents a huge area involving almost all the nodes of the network with a variety of congestion values. According to the semantic given to the signal representation, the comparison between them should produce a low similarity measure, which does not happen with a DTW-based comparison that assigns a similarity of 0.88.

1.6.2 Sampling

A comparison process always requires to identify the very representing properties of the considered objects, no matter what they are. In particular, when it comes to signals, the core of the comparison process is the sampling phase, during which some points of the signal are chosen to represent it. Since a signal is typically made by a huge number of points, in order to reduce the complexity of comparison sampling is necessary. In signal processing [84], sampling is used to transform a signal from continuous to discrete time. This process, by

choosing a suitable sampling rate, allows reducing the starting signal without loss of generality.

In our model, sampling is applied to the number of vehicles considered in the comparison rather than to time. By construction, choosing a fixed sampling rate could let us miss information about connected areas, or even completely loose the highest congestion values, and hence it could be not meaningful in order to represent the signal behavior. For this reason, we use a MinMax-guided sampling rate to compare different portions of signals, keeping track of the minimum and maximum congestion factor for each connected area. The following theorem shows that this sampling preserves the relative ordering between the congestion factors of the starting signal.

Theorem 1.6.1. *Let $f : V \mapsto [0, 1]$ be the function mapping a vehicle to a congestion factor, $\widetilde{f} : V \mapsto [0, 1]$ the same function after the sampling process, p_1 and p_2 two of the sampled peaks of the signal. Given u and v belonging to the connected areas corresponding to p_1 and p_2 respectively, the following holds:*

$$f(u) \geq f(v) \implies \widetilde{f}(u) \geq \widetilde{f}(v)$$

Proof. By construction, the peaks of the signal are ordered, so we have that $f(p_1) \geq f(p_2)$. We also know that, since p_1 and p_2 are sampled, their f and $\widetilde{f}$ are the same. Hence, we can infer that $\widetilde{f}(p_1) = f(p_1)$ and $\widetilde{f}(p_2) = f(p_2)$. Moreover, by signal construction, for any node

i in the connected area of p_1, $f(i) \leq f(p_1)$ holds. In particular, the inequality holds for $i = u$, hence $f(u) \leq f(p_1)$. Similarly, for any node j in the connected area of p_2, $f(j) \leq f(p_2)$ holds. In particular, the inequality holds for $j = v$, hence $f(v) \leq f(p_2)$. Thus:

$$\widetilde{f}(u) \leq \widetilde{f}(p_1) \geq \widetilde{f}(p_2) \geq \widetilde{f}(v)$$

The same result, with opposite ordering, would be obtained exploiting the minimal congestion values rather than the peak ones. □

1.6.3 Feature Extraction

The sampling phase leads us to the feature extraction phase. Exploiting the samples, for each connected area i of the signal, we identify the following parameters:

- $growing\_rate_i$, representing how the congestion factor grows in the connected area i, is computed as:

$$\frac{max_i - min_i}{|i|} \tag{1.6}$$

- $falling\_rate_i$, representing the change from the current connected area to the next one, is computed as:

$$\frac{max_i - min_{i+1}}{2} \tag{1.7}$$

In this case, we divide by 2 since the nodes involved in a high-low transition are always the peak of the current area and the starting element of the next one.

- $steady\_rate_i$, representing the number of nodes having the maximum congestion factor in the current area, is computed:

$$\frac{max_i}{|\{v \in i | c(v) = max_i\}|} \tag{1.8}$$

Once the parameters are fixed, the next step requires to establish how to use them for signal comparison.

1.7 Local Alignment: Motivations

Our last aim is not to align entirely the two signals, as DTW-based approach does, but we want to catch a meaningful similarity measure. With this purpose, for each area of the first signal we want to identify the best fitting area of the second one and perform an area-to-area comparison, according to the parameters extracted above. Thus, comparing signals simply following their natural evolution does not constitute the best option, in this case, especially since we cannot assume that the two signals have the same number of connected areas. For these reasons, before computing the parameters, we identify which area of the second signal should be compared with a given area of the first one, exploiting the local alignment strategy.

Once the algorithm, as reported in the preliminaries section, is applied, the returned alignment gives us a correspondence between areas of the two signals, suggesting us the best way to perform the comparisons.

To be more precise, we slightly change the local alignment algorithm, by considering not an exact match between the two sequences, but we introduce a tolerance η and we exploit the fact that our signals $signal_1$ and $signal_2$ to be compared have numeric values. Hence, we obtain the following *compare* function:

$$compare(i,j) = \begin{cases} ma & \text{if } |signal_1[i] - signal_2[j]| \leq \eta \\ mi & \text{otherwise} \end{cases}$$

1.8 Similarity Computing

Let n and m be the numbers of areas of the first signal and second signal, respectively, and let us suppose that, after the local alignment, we have to compare the area i of the first signal with the area j of the second one. We first compute (i) the parameters $growing\_rate_i$, $falling\_rate_i$, and $steady\_rate_i$ for the area i. The same for the connected area j; then (ii), we compute the differences between the areas with respect to each parameter:

- $d\_growing = |growing\_rate_i - growing\_rate_j|$;
- $d\_falling = |falling\_rate_i - falling\_rate_j|$;

- $d\_steady = |steady\_rate_i - steady\_rate_j|$.

Finally (iii), the similarity between the areas with respect to each parameter is computed as follows:

- $s\_growing = 1 - d\_growing$;
- $s\_falling = 1 - d\_falling$;
- $s\_steady = 1 - d\_steady$.

Iterating these three steps for any pair of areas to be compared, we obtain for each parameter as many values as the minimum between n and m:

$$s\_growing_1 \dots s\_growing_{min(n,m)}$$
$$s\_falling_1 \dots s\_falling_{min(n,m)}$$
$$s\_steady_1 \dots s\_steady_{min(n,m)}$$

A single similarity value for each parameter is extracted, through the following formula:

$$s_g = \frac{\sum_{i=1}^{min(n,m)} s\_growing_i}{max(n,m)} \tag{1.9}$$

A normalization is performed with respect to the maximum between n and m, in order to take into consideration all those areas of the

bigger signal that have not been considered in the comparisons. This allows discriminating those signals that, despite having some similar portions, have a very different number of connected areas. The formula 1.9 is computed in the same way for each parameter, obtaining s_f and s_{st} through $s\_falling$ and s_steady components.

The resulting s_g, s_f and s_{st} represent the similarity of the two signals with respect to the corresponding parameter. Now, we synthesize these three values in a single similarity measure. To this aim, we assign weights (α, β, and γ) to each partial similarity in order to highlight a behavior rather than another.

Definition 1.8.1 (Similarity Measure)**.** *Given α, β, and $\gamma \in [0,1]$ and such that $\alpha + \beta + \gamma = 1$, we define a similarity measure as follows:*

$$s = \alpha s_g + \beta s_f + \gamma s_{st} \tag{1.10}$$

Theorem 1.8.1. *Let $s = \alpha s_g + \beta s_f + \gamma s_{st}$ be the similarity measure, the following holds:*

$$s \in [0,1] \tag{1.11}$$

Proof. Let us recall that s_g, s_f and s_{st} depends on the values of $growing\_rate$, $falling\_rate$ and $steady\_rate$. In the first two cases, according to how they are computed in 1.6 and 1.7 respectively, their values are trivially in $[0,1]$, since they are obtained as difference between two congestion values (that are known to be in $[0,1]$). The

same holds for $steady\_rate$ whose value, according to 1.8, is obtained dividing a congestion factor by a value.

In order to prove that $s \in [0, 1]$, we first need to prove that $s_g \in [0, 1]$ too, and the same for s_f, and s_{st}. Let us focus on s_g: it is computed as in 1.9 summing $s\_growing$ values, which are trivially in $[0, 1]$, since they depend on $d\_growing$ obtained as difference of growing rates that we proved to be in $[0, 1]$. Hence, according to 1.9, the following holds:

$$s_g \leq \frac{min(n, m)}{max(n, m)}$$

which is obviously in $[0, 1]$. The same is clearly true for s_f and s_{st}. According to how s is computed in 1.10, given that α, β, and γ are chosen such that their sum is 1, and we have proven that s_g, s_f, and s_{st} are in $[0, 1]$, clearly $s \in [0, 1]$. □

1.9 Evaluation

In this section, some among the numerous simulations are shown, and also some of them highlight the difference between the proposed solution and the one obtained with dtw-based approach. For the local alignment algorithm, we chose the most commonly used values for match, mismatch and gap scores, namely 1, -1, and -2 respectively.

In the Table 1.1, we summarize the results of the experiments, obtained setting some fixed parameters the networks, namely the number

ex	s	dtw	r_1	r_2	sd_1	sd_2	η_1	η_2
1	0.53	0.92	10	15	10	10	0.5	0.5
2	0.8	0.95	10	15	8	12	0.5	0.5
3	0.31	0.9	10	15	10	10	0.2	0.5
4	0.66	0.91	7	15	10	10	0.5	0.5

Table 1.1: Simulation results with $\alpha = 0.5$, $\beta = 0.1$, $\gamma = 0.4$.

of nodes $n = 30$ and $m = 25$, and other variable parameters, such as the maximum allowed communication range r_1 and r_2, the standard deviation expressing the distribution of vehicles in the network sd_1 and sd_2, and the tolerance used for the local alignment η_a and η_b.

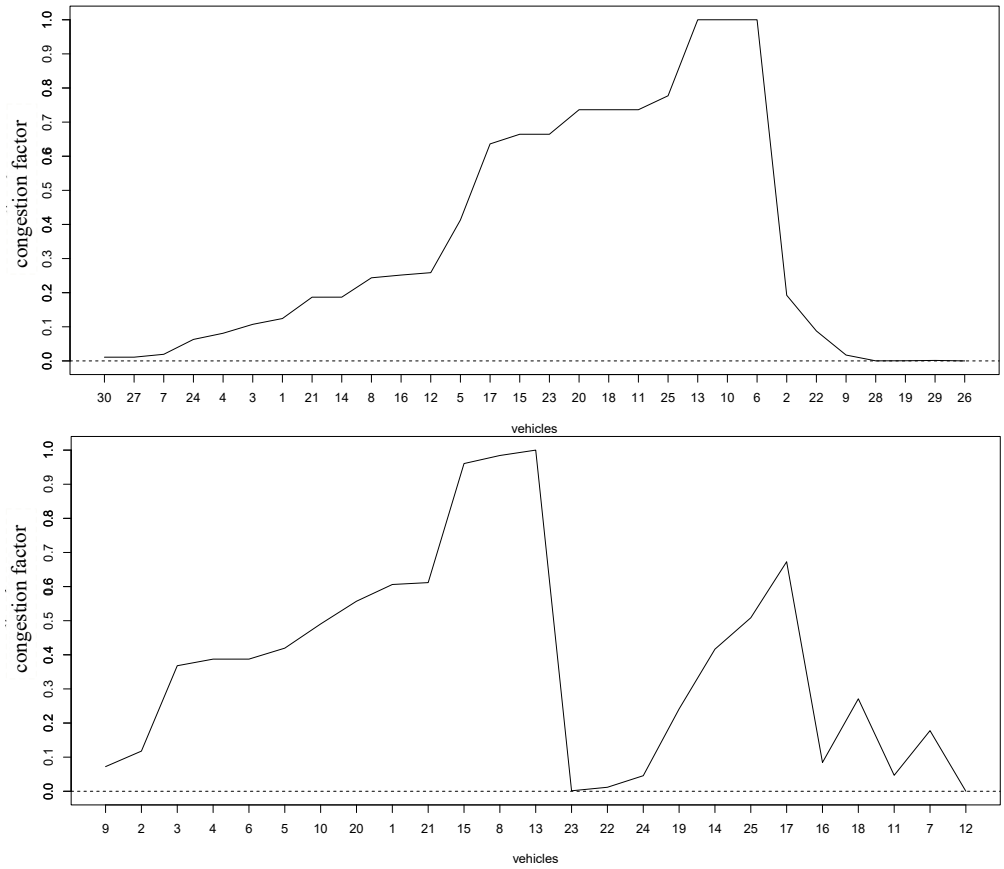


Figure 1.10: Signals corresponding to the experiment 1.

Looking at the Table 1.1, we can easily observe that the standard

dtw-approach leads quite often to very high similarity results. Our approach, instead, seems to take into account more precisely the parameters characterizing the behavior of the networks, leading to results that are quite more realistic. The example in Figure 1.10 shows that both the networks have 4 connected areas, but the way the nodes are distributed through these groups is very different. Indeed the first signal has just one area with more vehicles, while the other ones are made of just one vehicle. Differently, the second signal presents areas made of more than a single vehicle. This means that the similarity assigned by the dtw-approach is not correct, and does not catch this difference.

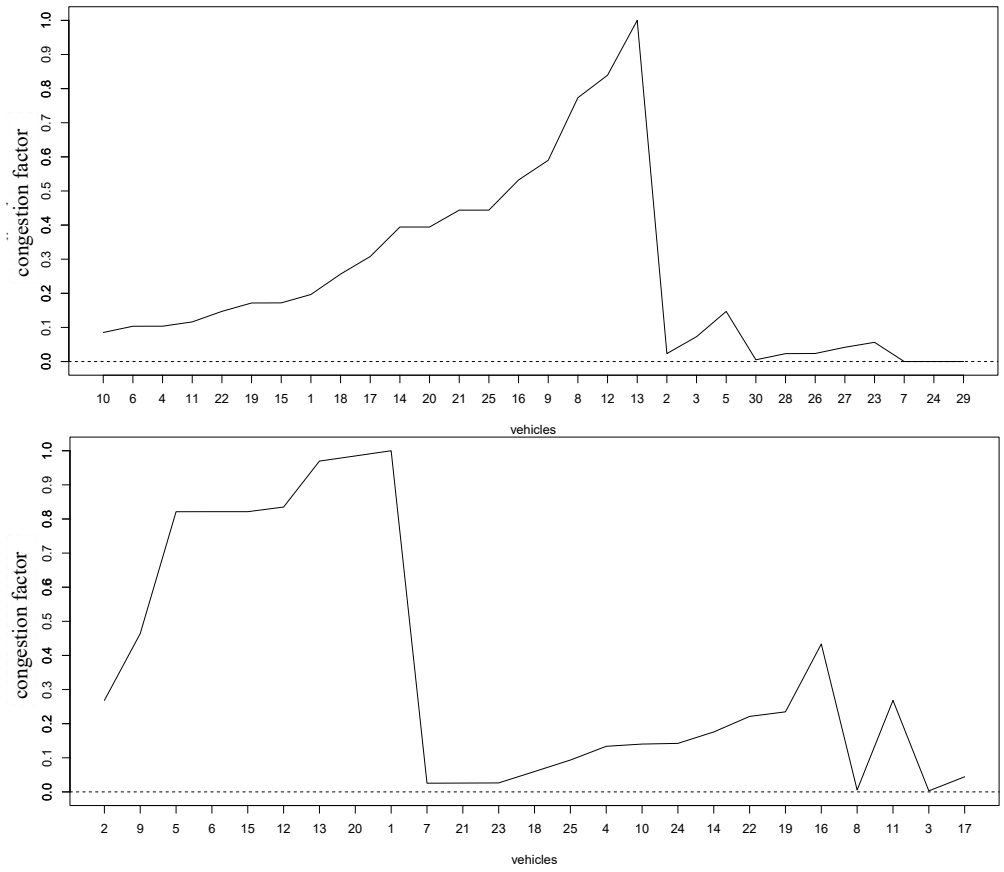


Figure 1.11: Signals corresponding to the experiment 2.

The second example, instead, in Figure 1.11, shows a first signal having 3 areas and 3 isolated nodes (meaning those nodes having zero congestion factor), and a second signal having 3 areas and 1 isolated node. In this case, the similarity produced by our approach is closer to the dtw with respect to the previous example. Indeed, these two networks present more affinities in the configuration.

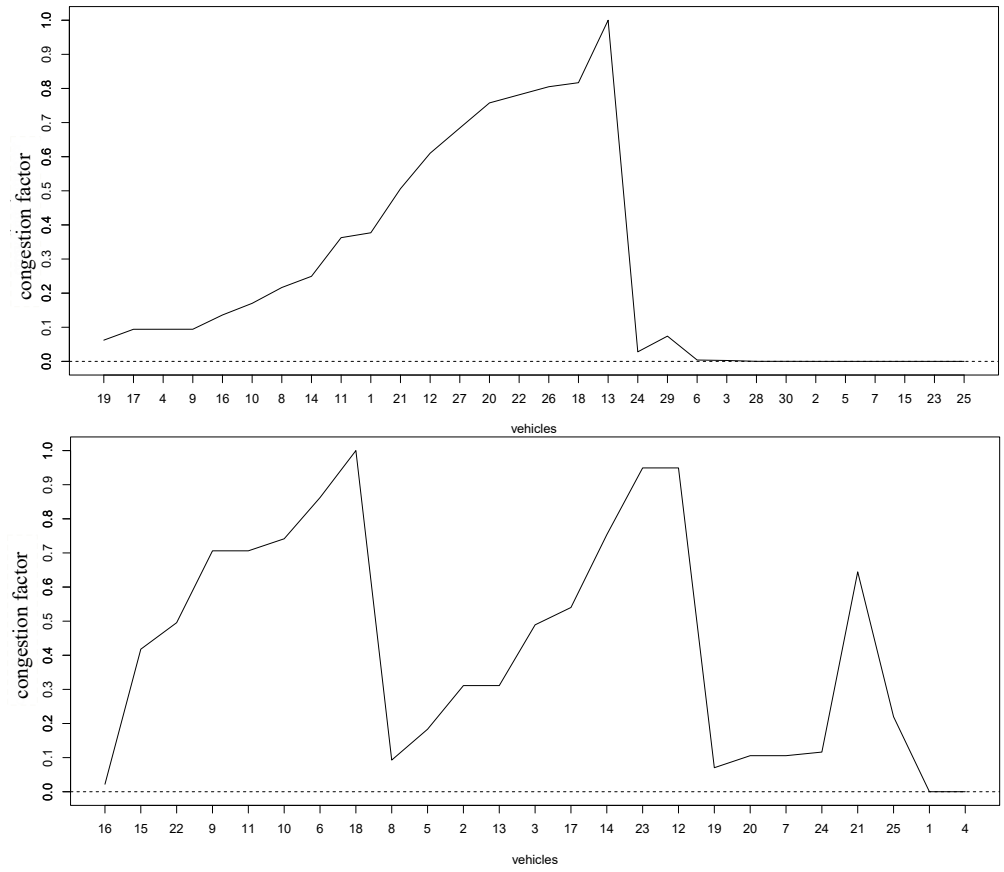


Figure 1.12: Signals corresponding to the experiment 3.

In the third example of Figure 1.12, still the networks have the same number of areas, in this case 4, but we can easily observe that they are very different. Indeed, the first network present many isolated nodes, and almost all the remaining nodes fall into the first area. Conversely,

the second network presents 3 areas made of quite the same amount of nodes, and 1 made of a single node. The difference in the configuration makes our approach produce a low similarity, differently from the dtw-method.

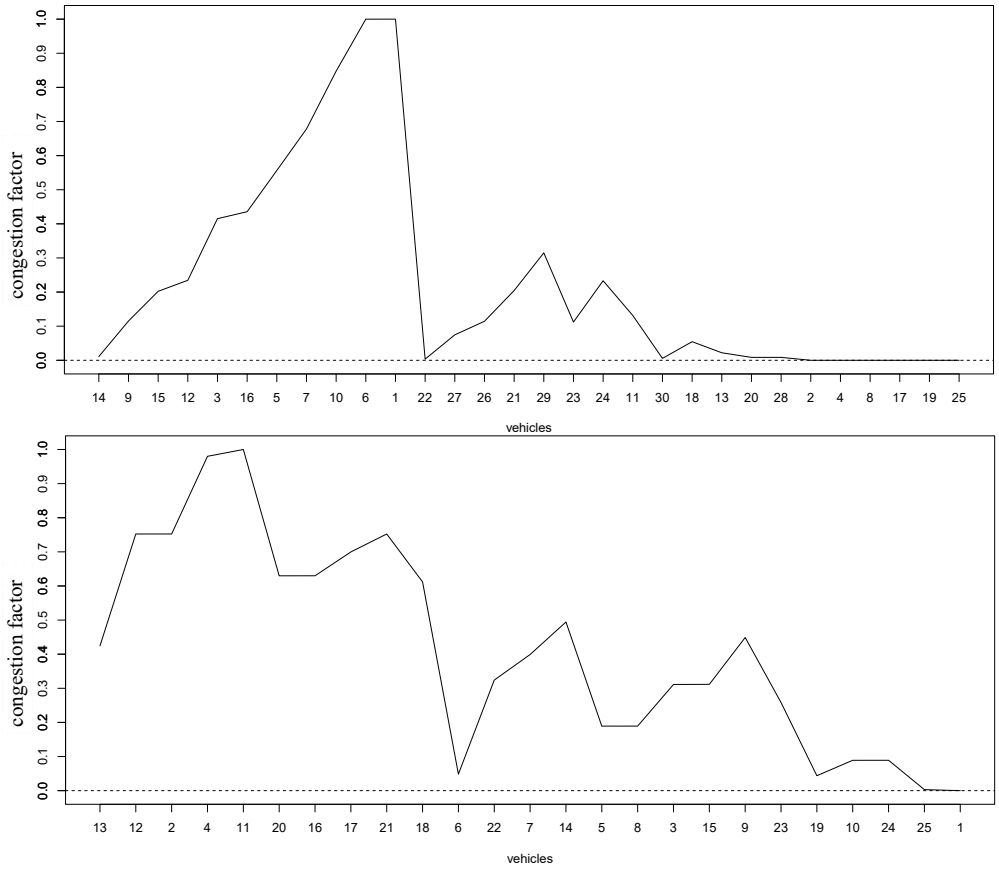


Figure 1.13: Signals corresponding to the experiment 4.

Finally, in the fourth example of Figure 1.13, the first signal is made of 8 areas, while the second one of 7. They present some affinities in the left part of the signal, while the right one where isolated nodes appear is very different. Instead, the result says that these two signal are similar for a little bit more than the half, which is quite reasonable looking at the graphics.

It is important to point out that after the alignment is made, we need to run as many comparisons as the smallest size of the compared signals, which asymptotically corresponds to $\mathcal{O}(min(n, m))$, hiding a constant due to the introduced weights.

Chapter 2

Algorithms for Smart Parking

2.1 Introduction

Nowadays, the Internet of Things (IoT) [61] is becoming more and more integral part of our lives. Just think of autonomous vehicles [135], smart cities [60], and also autonomous vehicular networks [9]. This trend is supposed to continue if we just have a look at the study reported in [122], where they predict that, by the year 2050, almost the 65% of population will be living in the city. This phenomenon would clearly increase the number of vehicles along the city roads, by highlighting the relative problems, consequently: starting from the traffic congestion, according to the study of [121], Rome is on the top of the ranking of the cities with the longest traffic jam delays in Europe,

but also the gas emission [110], and the high demand of parking space.

Parking slot detection is a challenging issue in vehicular ad hoc networks, since it is a process that seriously impacts several aspects. First of all, drivers looking for a free parking slot cause most of traffic congestion due to the need of travelling the same roads over and over until an available slot is found. Moreover, also the stress suffered by the drivers has to be taken into account: very often people come to give up on a dinner if they only think of the stressing situation of looking for a free parking slot. Last but not least, environmental pollution. As pointed out in [65], vehicle energy consumption and the urban air quality need to be assessed. They also specify that the total air pollution in urban areas is due to exhaust emissions from traffic flow.

The parking problem can be investigated from two main perspectives:

- One can provide a centralized solution, like they did in [116], where a way to assist users to get parking information by relying on a web application is given.

- But one can also prefer a decentralized approach, like [118], that aim at maximizing the autonomy of drivers, without any intermediaries, which in our opinion is more suitable for the problem we are facing.

In [77], they provide an interesting classification of the smart park-

ing solutions from the literature, by individuating three macro-categories:

- Information Collection includes those solutions relying on sensing techniques to identify parking status.
- System Deployment is about software and statistical analysis on collected data.
- Service Dissemination deals with the relationship between information and social features, such as the competition that normally happens between two drivers contending the same slot.

The proposals provided in this thesis lay on the third category since it is explicitly taken into consideration the competitive nature of the parking process. Moreover, the preferred approach is the decentralized one, so to preserve the self-organizing capability of VANETs.

In the next sections, two proposals for smart parking will be provided. The first one relies on Ant Colony Optimization approach to push drivers choosing a path rather than another creating an isomorphism between driver's behavior and real ants one while looking for food. The main idea is to use the pheromone, which in real life attracts ants along a path, as a repulsive so to avoid roads being crowded. The second proposal exploits the power of blockchain consensus mechanism to guarantee a fair vehicle distribution over a consortium of car parks. Precisely, all the operations that modify the network configuration have to be accepted in advance by the competitors (namely the

members of the consortium) so to force the competitors themselves to behave fairly.

2.2 Related Work

Smart parking is one of the most challenging application of vehicular ad hoc networks. Finding available slots either in urban areas or parking may cause traffic congestion, environment pollution, and impact drivers mood. For these reasons, in the last few years, this topic has become a very hot research topic and an increasing number of researchers use known existing algorithms and adapt them to face the parking problem. The literature is made of several solutions aimed at handling the traffic congestion caused by drivers looking for an available parking space. Several authors rely on Ant Colony Optimization, (ACO, for short) to find a solution for the parking problem: for instance, in [133], they apply ACO to solve the parking problem, but in a totally different manner, since they focus on autonomous vehicles. See [52] for a survey on ACO. It provides an interesting way of approaching the solution of a problem, as it follows the biological behavior of living beings: indeed, it has not only been used for parking applications, but also in robotics, such as in [132], where it is applied to avoid collisions with an obstacle while programming the movement of a robot.

Other interesting solutions on smart parking are provided in [24], where authors provide a logic approach to the distance geometry prob-

lem (DGP, for short), by classifying vehicles in *ready*, *finding*, and *leaving*, according to what is their role in the current network: starting from the network configuration, they draw an indirect graph representing all the parking information related to each vehicle, and transmit them via V2V communication. There are also machine-learning-based approaches providing a smart parking system, such as the one proposed in [115]. Moreover, authors in [8] proposes an interesting framework based on a deep long short term memory network and a way to predict the parking availability.

The combination between the blockchain technology and VANET environment has been investigated already, since it provides a self-organizing, decentralized and transparent system, such as in [74], where authors rely on *Ethereum*, a decentralized platform based on Bitcoin's blockchain concept, and they propose mandatory applications such as traffic regulation application, vehicle tax and vehicle insurance applications. In [117], authors consider a new type of blockchain which aims to improve critical message exchange in VANET, via the creation of smaller blockchains for country-level local communication and a larger "public" one which stores trusted nodes. Authors of [82] highlight that security in VANET is still an open issue, due to its features: for this reason they provide a blockchain-based anonymous reputation system (BARS) in order to establish a privacy-preserving trust model for VANETs, by using a reputation algorithm that prevents the distribu-

tion of forged messages. Furthermore, authors of [137] also investigate security issues in a 5G-VANET: they determine that using a blockchain in such a network allows detecting malicious nodes and messages, significantly improving the trustworthiness of the whole network, while the performance remains acceptable. Another interesting proposal is [138], where authors propose a proof-of-event consensus algorithm for blockchain, rather than the more common proof-of-work: data are collected by the road side units and every time a vehicle receives an event notification it has to verify the correctness.

Many authors also investigate the parking problem in particular, by dealing with it through blockchain. Precisely, authors of [5] and [66] propose a schema to preserve driver's privacy; similarly, authors in [141] propose ParkChain, which is a smart parking system based on blockchain, aimed at guaranteeing that no one can delete, revert, hack or question the time a registered vehicle securely entered a parking area.

2.3 Ant-Colony-Optimization-Based Parking Algorithm

For the first proposal, it has been studied the solution to a well known optimization problem, based on the behavior of real ants in nature, the ant colony optimization problem (ACO, for short), with the goal

of applying such a solution to the parking problem in a decentralized manner. In particular, the approach takes inspiration from ACO, but following an opposite direction: indeed, while in ACO ants are attracted from paths with a higher amount of pheromone, which is the chemical substance released from an ant that followed the same path previously, in the considered scenario the pheromone acts as a repulsive for drivers, so to avoid crowded situations.

More specifically, anytime a driver follows a path P, the associated pheromone will be updated so to make other drivers understand that P could be a potentially crowded path. In this way, anytime a driver has to choose a path, by following the pheromone it is guaranteed that the less crowded one is selected. The final result is a context-aware and self-organizing network, characterized by an even distribution of vehicles among the available parking slots, with a lower gas emission due to the multiple tours that drivers usually have to perform to find a space.

2.3.1 Preliminaries

Environment Classification

To deal with parking problem through ACO, the environment is structured as a huge area made of parking slots (taken or available). Without loss of generality, we assume that the environment is divided into regions, such that each region has a capability in terms of available

parking slots.

Such an environment can be represented as a directed graph $G = (V, E)$, where each vertex $v \in V$ is a region with $|V| = n$, and each edge $(i, j) \in E$ is a connection between regions, such $(i, j) \in E$ if and only if from the region i the region j is reachable directly. Moreover, a weight function $w : V \times V \longmapsto \mathbb{N}$ associates to each edge a weight given by the capability of the destination node.

About Ant Colony Optimization

As precisely explained in [52], Ant Colony Optimization was introduced in 90's as a nature-inspired metaheuristic for the solution of hard combinatorial optimization problems. It follows the behavior or real ants looking for food: in the beginning, they visit the neighborhood of their nest at random, they find some food source and get it back to the nest. During the trip, ants release a pheromone trail to guide other ants towards the food.

The Ant Colony Optimization (ACO) is a genetic algorithm that allows you to solve the Traveling Salesman Problem (TSP) that requires to find the minimum costing path to reach a specific node in the graph visiting every node at most once. In the ACO, every agent participates individually in the construction of the solution, building, iteratively, different candidate solutions for the TSP that converge, step by step, in a single one thanks to a shared memory system which

allows each agent to share with the others information relating to their experience about the choice of a certain direction rather than another.

The ACO Parameters The main parameters of ant colony optimization paradigm are the following [85]:

- η_{ij} is the attractiveness of the move from i to j, according to a priori information;
- τ_{ij} is the trail level of the move from i to j, according the a posteriori information, given by previous iterations.

At any iteration, the ant k performs a move from i to j with a certain probability p_{ij}^k that depends on attractiveness and trail level values. The probability is computed as follows:

$$p_{ij}^k = \begin{cases} \dfrac{\tau_{ij}^\alpha + \eta_{ij}^\beta}{\sum_{(ij)\notin tabu_k}(\tau_{ij}^\alpha + \eta_{ij}^\beta)} & \text{if } (ij) \notin tabu_k \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where:

- α and β are parameters between 0 and 1 that allow to specify how determinant are the trail level and the attractiveness, respectively;
- $tabu_k$ is the list of not feasible moves for the ant k starting from the state i.

Once any ant completes its move, the trail level at iteration t is updated according to the trail level at the previous iteration, as follows:

$$\tau_{ij}(t) = \rho * \tau_{ij}(t-1) + \Delta\tau_{ij} \tag{2.2}$$

where: ρ is a parameter representing the *evaporation rate* that guarantees a way to avoid too fast convergence of the algorithm to a suboptimal region, allowing the exploration of new areas of the search space [51]; $\Delta\tau_{ij}$ is the pheromone, meaning the sum of the contributes given by all the other ants that have performed the ij move already.

The ACO Algorithm Typically, an algorithm following ant colony optimization is based on the repetition of three phases [51] until a termination condition is reached, as shown in Figure 2.1.

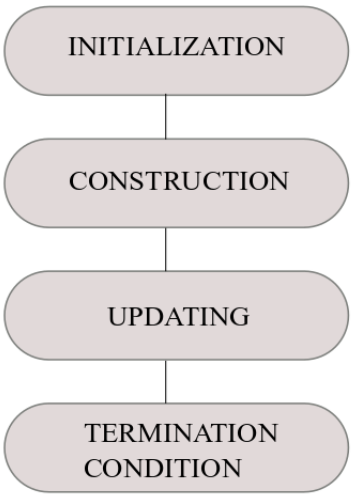


Figure 2.1: Phases of a typical algorithm solving a problem according to ant colony optimization

The *initialization* phase consists in assigning a starting value to

trail level and pheromone parameters, for any edge of the graph.

The *construction* phase is about the colony of ants, each of them computing its own solution according to a stochastic local decision policy, based on pheromone and trail level.

The *updating* phase is the moment when trail level and pheromone are modified: in particular, the trail level can either increase or decrease according to if ants deposit pheromone or not.

Isomorphism of the Problem

To formulate the optimization problem that represents the situation where a driver moving toward a destination wants to find an available parking slot, we need to take into consideration:

- The position of the driver;
- The position of the destination.

Clearly, the ideal situation would be finding the parking slot which is the closest one to the destination point. The first approach is to solve the problem by finding a path, from the position where the driver is to the one of the place he wants to reach, that maximizes the number of available parking slot met, so to increase the chance that the driver completes the parking process. Figure 2.3.1 shows an example of a graph made by five regions connected according to the parking slots available in the destination region, while Figure 2.3.1 shows the path

that maximizes the number of available parking slots met, assuming that the vehicle start from region A, and wants to reach region E.

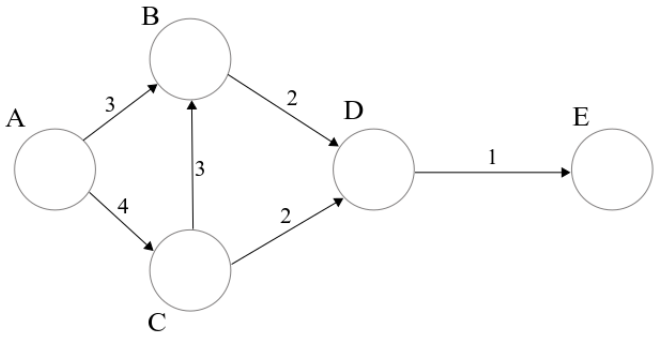


Figure 2.2: Example of a graph of five regions and corresponding weighted connections

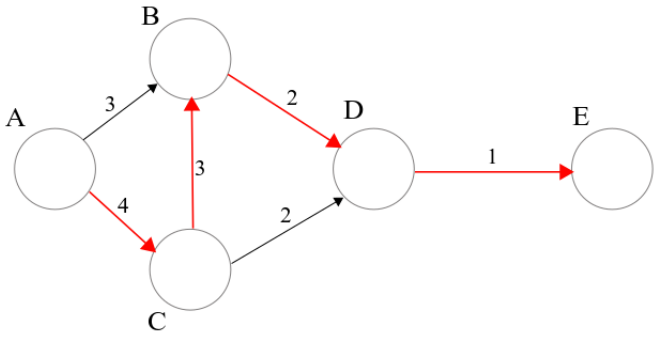


Figure 2.3: Example of the chosen maximizing available parking slot path assuming A and E as starting and ending point respectively.

Formally, let w_{ij} be the weight of the edge $i \to j$ in the graph G, and let P be the resulting path of maximum weights, the optimization

problem can be formulated as follows:

$$\begin{cases} \max \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} * x_{ij} \\ \sum_{i=1}^{n} x_{ij} \leq 1 \\ \sum_{j=1}^{n} x_{ij} \leq 1 \\ P[1] = S \end{cases} \tag{2.3}$$

where x_{ij} can assume values in $[0, 1]$ according to the resulting path P:

$$x_{ij} = \begin{cases} 1 & \text{if } i \rightarrow j \in P \\ 0 & \text{otherwise} \end{cases}$$

The first inequality constraint guarantees that each node of the resulting path has at most one outgoing arc (except for the final region that has not outgoing arcs). Similarly, the second inequality constraint ensures that any node has at most one incoming arc (zero in case of the starting region). Notice that in the optimization problem shown in (2.3), it is required that the starting region appears in the resulting path (as first region precisely), but this does not happen for the final one: indeed, if the destination region has not available slots, any edge leading to it has a null weight and hence it is not chosen as the optimal path.

2.3.2 Towards ACO for Parking

The problem, as defined so far, does not take into consideration the actual vehicle distribution over the network. It is clear that there exists a mapping from the parking problem to an ant colony optimization one, but it is important not to loose the real-time nature of VANETs. For this reason, in the following, a more complex model of the netwrok is provided.

2.3.3 Model Definition

The model configuration can be seen in Figure 2.4. In the proposed setting, it is assumed that each driver, that has to park his car, has a starting region that is known, and a destination region towards which he wants to get as close as possible. Hence, the graph configuration depends on a fixed destination region for any driver taking part of the parking process.

As shown in Figure 2.4, for each parking region of the model, and each edge connecting nodes, some properties are provided:

- For nodes:
  - w: the distance to walk to reach the destination region;
  - a: the number of available parking spots in the region.
- For edges

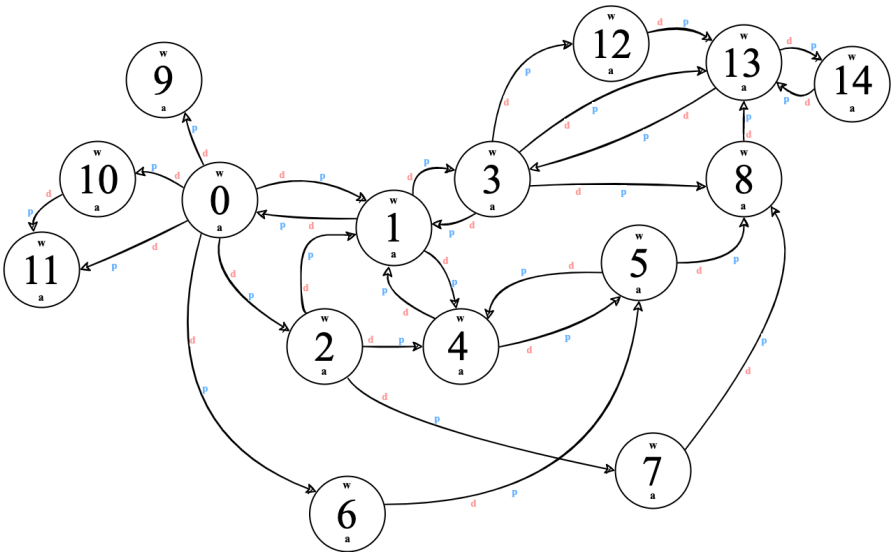


Figure 2.4: Parking regions graph.

- d: the distance to travel by road to reach the destination node of the arc from the source one;
- p: the probability with which each vehicle will visit the destination node of the arc from the source one; since it is a probability, it is a value between zero and one. As in the standard ACO, this parameter works as the pheromone but, differently form ACO, it has a repulsive power, rather than an attractive one.

2.3.4 Graph Coloring

In order to avoid drivers choosing the wrong edges, a coloring mechanism is provided. "Wrong edges" means edges that push the driver irreversibly away from the destination, rather than getting him closer to it. Since, at the very beginning of the execution, there is no pheromone

yet to inform drivers of which edges should be picked and which should not, a graph coloring is needed to prevent wrong choices that would lead to a bad exploration of the graph. If we consider that the starting node is the 0 one and the destination node is the 8 one, at the beginning of the execution of the algorithm, without considering the graph coloring, the vehicles will have to do the following evaluation to choose the node to move to, by maximizing the ratio $\frac{a}{d*w}$. Such a ratio is:

- $\frac{4}{1.6*1.16} = 2.16$ for the node 1,
- $\frac{3}{1.2*1.5} = 1.67$ for node 2,
- $\frac{7}{2*3} = 1.17$ for node 6,
- $\frac{16}{2.4*3.4} = 130.56$ for node 9,
- $\frac{21}{0.8*2.8} = 9.38$ for node 10,
- $\frac{4}{1.3*3.34} = 0.92$ for node 11.

With these values, the node 9 would be chosen, but it would bring the vehicle irreversibly far from the destination.

In order to avoid such a problem, the graph is previously colored as in Figure 2.5, so that every agent is aware of which node can bring it to the destination, or can get him closer to it, at every step. Precisely, a parameter θ is defined, which indicates the maximum distance that is reasonable to walk from the parking slot to the destination:

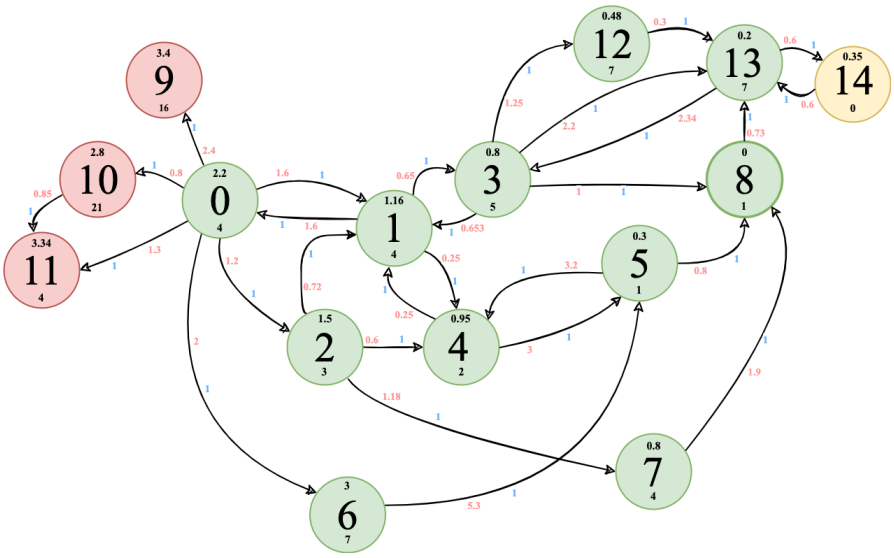


Figure 2.5: Colored parking regions graph assuming that the destination is the node 8.

- The red nodes are those that do not lead to the destination node, and hence they should be avoided;

- The green nodes are the ones that bring to the destination or to a parking region within the distance θ;

- The yellow is associated to those nodes v having only one outgoing arc, which is directed only to a node having v among its adjacent nodes. Such a configuration might be source of annoying loops, and for this reason it is imposed that yellow nodes should be visited only once by each vehicle, unless a parking slot is made available in the region: in this last case, a further visit is allowed to complete possibly the parking process.

Clearly, as in the standard version of ACO, at the beginning, the graph does not provide a meaningful pheromone information, indeed

it has value 1 on every edge. In this phase, the edge chosen by the driver, among the available ones with the same amount of pheromone, is the one that maximizes the ratio $\frac{a}{d*w}$, where a, d, and w have been defined previously.

2.3.5 The Pheromone Model

In the proposed algorithm, in order to avoid situations where all vehicles follow the same path, the pheromone causes a repulsion instead of attraction and it is modified as described below:

- decrease of p value: when a vehicle travels on an arc, the p value of the entering arcs of the node to which it has moved are decreased by:

$$\frac{(d * w)}{(a + 1)} * 10^{-1,9} \tag{2.4}$$

  The decreased value is deliberately lower to the increased one in order to avoid losing information too quickly.

- increase of p value: when a vehicle leaves a region or when it parks in a region, the p value of the entering arcs of the node is increased by:

$$\frac{a}{(d * w)} * 10^{-2} \tag{2.5}$$

  The increase is proportional to the available parking slots in the region in which the driver has parked (or has leaved) and inversely proportional to the distance to travel by road and the

one to walk to reach the driver destination node;

- total evaporation: a the end of each step (every vehicle has made a movement) an update of p of each arc is made so that those arches that have been visited previously attractive again:

$$\frac{1}{d * 1000} \tag{2.6}$$

this p increase on each visited edge is inversely proportional to the length of the edge in order to disadvantage the choice of longer ones.

where the value of w for the destination node is not equal to 0 but to 0.02 considering that also a parking slot in the destination region is not the place where the driver is headed.

2.3.6 Constraint Relaxation

In a realistic scenario in which there are few available parking slots, it is not always possible to guarantee that all vehicles will reach the destination or an acceptable node (according to the distance θ) without visiting a node they have already visited; for this reason, a process of relaxation of the constraint on the θ value is needed in order to facilitate the choice of the region in which allocating the car.

More precisely, the relaxation of the constraint on the acceptable walking distance θ, which consists in increasing the distance θ by 0.5,

is applied every time a vehicle visits a node that has already been visited previously; in this way, for example, also the red nodes could be considered for allocation (desperate times call for desperate measures).

Through such a trick, unwanted loops are handled, and it is guaranteed that for each driver a solution will be eventually found.

2.3.7 The ACO Algorithm for Parking

Every algorithm that follows ant colony optimization is based on the repetition of "Initialization", "Construction" and "Updating" until a "Termination condition" is satisfied.

Before showing the algorithm, let us first recall some parameters:

- D is the drivers population in which every driver has a position, a θ and a list of positions history to keep track of the nodes he has already visited;
- E is the set of edges of the parking regions graph.

After coloring the graph and initializing the drivers population D, as long as there is a driver who has not parked, each driver will always move, step by step, through the edges that maximize the pheromone or the ratio $\frac{a}{d*w}$ ($findNextNode$). Every movement of the drivers results in a decrease and an increase in the pheromone on the edges:

- increase of pheromone: the incoming edges pheromone of a parking region is increased when a driver park or leaves that parking

Algorithm 1 optAco

```
1: Color the graph
2: for each d in D do
3:     set d → pos equal to start_node
4:     set d → θ equal to 0.4
5: end for
6: repeat
7:     for d in D do
8:         if d not parked then
9:             n_node ← findNextNode(d → pos)
10:            if n_node ≥ 0 then
11:                increaseIncomingEdgesPh(d → pos)
12:                moveToNode(d, n_node)
13:                decreaseIncomingEdgesPh(d → pos)
14:                if d has already visited n_node then
15:                    relax Walk Distance Constraint(d)
16:                end if
17:                if n_node → w ≤ d → θ ∧ n_node → a ≥ 0 then
18:                    park d in n_node
19:                    increaseIncomingEdgesPh(d → pos)
20:                end if
21:            end if
22:        end if
23:    end for
24:    for e in E do
25:        Pheromone_e− = Evaporation
26:    end for
27: until every d parked
```

region;

- decrease of pheromone: the incoming edges pheromone of a parking region is decreased when a driver move into it;

Each position taken by the driver is collected in a list so that, if a node is visited more than once by the same driver, the relaxation on θ constraint can take place.

2.3.8 Valuation

In this section, some simulation results are provided from the execution of the defined algorithm, as well as the behavior of the algorithm is analyzed by changing the number of drivers, or the destination node.

2.3.9 Simulation Results

Considering the graph in Figure 2.5, if the starting node is 0, the driver population has a cardinality of 15, the path of nodes of the graph covered by each agent to reach the destination is shown in the Table 2.1.

If we consider that at the step 16 parking spots is left and available again in node 14, path of the drivers is shown in Table 2.2

Table 2.1: Driver path with always 0 available parking spots in 14

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_10
D_1	0	1	3	8						
D_2	0	2	7	8	13					
D_3	0	1	3	8	13					
D_4	0	2	7	8	13					
D_5	0	1	3	8	13					
D_6	0	2	7	8	13					
D_7	0	1	3	8	13					
D_8	0	2	7	8	13					
D_9	0	1	3	8	13	3				
D_{10}	0	2	7	8	13	3	8	13	3	
D_{11}	0	1	3	8	13	14	13	3		
D_{12}	0	2	7	8	13	3	8	13	3	
D_{13}	0	1	4	5						
D_{14}	0	2	7	8	13	3	12	13	14	13
D_{15}	0	1	4	1	4					

2.3.10 Benchmarks

Some observations on the behavior of our algorithm have also been performed, by starting from the Figure 2.5, to check the number of steps needed to complete the parking process, by first varying the number of drivers (Table 2.3), then the destination nodes (Table 2.4).

The columns of Table 2.3 represent the number of drivers, while the rows are the number of steps needed on average to complete the parking process. The columns of Table 2.4 represent the destination node, while the rows are the number of steps needed on average to complete the parking process.

Table 2.2: Driver path with a new available parking spots in node 14 from step 16

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	
D_1	0	1	3	**8**					
D_2	0	2	7	8	**13**				
D_3	0	1	3	8	**13**				
D_4	0	2	7	8	**13**				
D_5	0	1	3	8	**13**				
D_6	0	2	7	8	**13**				
D_7	0	1	3	8	**13**				
D_8	0	2	7	8	**13**				
D_9	0	1	3	8	13	**3**			
D_{10}	0	2	7	8	13	3	8	13	**3**
D_{11}	0	1	3	8	13	14	13	**14**	
D_{12}	0	2	7	8	13	3	8	13	**3**
D_{13}	0	1	4	**5**					
D_{14}	0	2	7	8	13	3	12	13	**3**
D_{15}	0	1	4	1	**4**				

Table 2.3: Step needed varying the number of drivers

	5	10	15	20	25
Average number of steps	3.8	4.2	5.07	5.7	6.56

Table 2.4: Step needed varying destination node

	3	8	12
Average number of steps	3.1	4.2	3.7

2.4 Competitive-Blockchain-Based-Parking System with Faireness Constraints

The second proposal for smart parking algorithm is developed in a different scenario. Precisely, a consortium of car parks, rather than a single one, is taken into consideration.

Let's imagine having an autonomous parking system able to allocate vehicles approaching several car parks, in such a way to make drivers happy about the slot chosen for them, with regards to the position. It is quite reasonable speaking about several car parks when we have to deal with parking process in a wide urban area. Just think of the numerous car parks close to some shops or markets. Most probably, one of them is in a very strategic position, close to the most famous shops or interesting zones. Clearly, it would be the best choice for any driver looking for a free slot, but also the best choice for any allocation algorithm based on the smallest distance between the chosen slot and the destination of the driver. This scenario clearly demonstrates that if all the drivers try to reach the same car park, their vehicles would create even more traffic congestion, instead of reducing it.

Using a consortium of car parks (which are in the same neighborhood) able to guarantee a fair distribution of vehicles, by choosing, when it is possible, a slot which is near to the destination of the driver, it is possible to dramatically reduce the traffic congestion caused by

parking. By moving the attention from the single parking area to a consortium, we make clear that the vehicle allocation problem does not affect the driver or the parking owner alone, but it should be handled in the best interest of all of them. Introducing a consortium requires rules and constraints to manage it. To this aim, it has been decided to use the philosophy behind the blockchain mechanism based on a consensus to be reached before any operation on the network is performed. In our case, prices variations and capacity constraint modifications from any of the members should be approved, in a certain percentage, by any other member for a fair competition, aimed at an even distribution and not at the enrichment of the single.

It will be shown how, guaranteeing a fair competition among competitors and an even vehicle distribution among them, the solution provides drivers a secure parking mechanism, which only aims at allocating vehicles in the smartest way possible, no matter what the economic interests of the single competitor are. Moreover, environment, cities, and citizens are going to benefit from the blockchain integration.

2.4.1 Background

Let us recall some notions that will be used in the sequel: the destination-based algorithm used to allocate vehicles in the available parking slots, which is based on some known algorithms for memory allocation in operating systems, and also the basics of blockchain technology and what

are the aspects that can be exploited in a parking process and why it is worthy.

Distance-based parking algorithm

Before introducing the blockchain-based parking system, the algorithm that is used is the distance-based parking algorithm proposed in [19]. The main idea is to reduce the vehicle parking problem to the most general and commonly known memory allocation problem in operating systems.

The central memory is a sequence of blocks of variable size, where processes, in order to be executed, need to be loaded beforehand.

The isomorphism between the two problems is quite intuitive: the set of vehicles ready to be parked, called *ready queue*, corresponds to the processes ready for the execution, the parking area corresponds to the central memory, and the time during which the vehicle will occupy the slot corresponds to the execution time characterizing each process. Similarly, dynamic memory allocation methods are customized and used. Let us recall the basic versions: (*i*) First-Fit, the first available and big-enough block of memory is chosen; (*ii*) Best-Fit, the smallest block, among the available ones and the ones having sufficient size, is chosen; (*iii*) Worst-Fit, the biggest block, among the available ones and ones having sufficient size, is chosen. In the sequel, these policies (First-Fit, Best-Fit, Worst-Fit) will be mapped into the parking

domain.

In memory allocation the main goal is to optimize the resources, while in the parking one the goal is to fill the parking area as soon as possible, which means trying to consume the ready queue in the minimum time possible. Consequently, it is possible to define the objective function formally as follows:

Definition 2.4.1. *Objective Function*
The objective function for the parking process requires to minimize the time to make the ready queue empty.

$$min_{filling} is\_empty(ready\ queue) \tag{2.7}$$

The objective function expressed as in Definition 2.4.1, says that the algorithm is going to minimize the time needed to find a slot for any vehicle in the *ready queue*: from the vehicle point of view, the total wait time will be reduced.

Each parking area is divided into slots. Each slot is represented by its position and its size. An example of values associated to each slot of each parking area is shown in Table 2.5.

The most innovative aspect of the proposed algorithm is that the *distance* parameter is used. Specifically, we speak about the distance between the parking slot and the final destination of the driver (infor-

Table 2.5: Example of information about slots position and size.

	$slot_1$	$slot_2$	$slot_3$	$slot_4$
$Area_1$	(5,10)	(6,11)	(4,6)	(3,8)
$Area_2$	(2,12)	(3,12)	(4,8)	(5,8)
$Area_3$	(3,11)	(4,10)	(2,5)	(1,5)
$Area_4$	(1,10)	(2,3)	(3,10)	(4,5)

mation which is supposed to be known in advance). By using such a parameter, and assuming that the vehicle v has to be parked, the typical approaches for dynamic memory allocation in operating systems are revisited as follows:

- **Revisited-First-Fit (RFF)**: among all the available parking slots able to contain v, the nearest to the destination is chosen;
- **Revisited-Best-Fit (RBF)**: considering the size of the vehicle, among all the available parking slots which are large enough for the vehicle, the closest and smallest one is chosen;
- **Revisited-Worst-Fit (RWF)**: taken in consideration the size of the vehicle, among all the available parking slots with dimension equal or greater than the vehicle, the closest and biggest one is chosen;

Simulation results prove that the revisited versions of First, Best, and Worst-Fit reduce the waiting time for the ready vehicles, and hence the total time needed to empty the queue. In Table 2.6, experimental results obtained after 4 blocks of 200 executions for the same parking

area are reported. Every time we generate 11 vehicles with random features. The term "positive simulations" means a simulation in which the behavior of the revisited algorithm is better than the standard one. In Table 2.7, instead, failed attempts are provided. It's easy to see that the worst-fit presents more failed attempts than the other ones, but still with very small values with respect to iterations number.

Table 2.6: Positive simulation results for the revisited versions of First, Best, and Worst-Fit.

	RFF	**RBF**	**RWF**
FirstBlockSuccess	11	21	18
SecondBlockSuccess	128	17	24
ThirdBlockSuccess	124	17	23
FourthBlockSuccess	121	19	27

It is also summarized, in Table 2.8, the comparison results among the standard algorithms and the revisited ones: in particular, the revisited First-Fit yields an improvement around the 60%.

Table 2.7: Negative simulation results for the revisited versions of First, Best, and Worst-Fit.

	RFF	**RBF**	**RWF**
FirstBlockFailure	/	/	1
SecondBlockFailure	/	/	1
ThirdBlockFailure	/	3	1
FourthBlockFailure	/	/	/

Table 2.8: Improvement rate for the revisited versions of First, Best, and Worst-Fit.

	RFF	**RBF**	**RWF**
IR	60.5%	9.3%	11.5%

Blockchain

Blockchain has been introduced in 1991 and it is an emerging technology which is changing the way new systems operate: it is based on unifying a ledger between all the nodes of the network in a distributed manner, by avoiding that other nodes join the network without authorization [95]. A blockchain consists of a chain of blocks, where each block contains data whose access is handled through cryptography (typically the hash code of the current block, but also the code of the previous one). Anytime someone joins the network, he receives a copy of the blockchain: when someone adds a new block, it has to be validated from the rest of the network. As clearly explained by authors of [47], each transaction in the public ledger is verified by consensus of a majority of the participants in the system.

Due to the numerous potentialities, blockchain are used for very different applications, not only in financial fields, but also non-financial ones, such as health, music, and industry [71, 13, 34].

The typical use of blockchain is for financial purposes: if A wants to send money to B, such a transaction is represented as a block and

submitted online for validation. Only after the transaction is approved by the network participants, the corresponding block is added to the chain, and finally A sends money to B [47].

The name *blockchain* is very explicative about how this technology works: the basic mechanism is a consensus validation algorithm, which allows a peer-to-peer communication among participants, without the need of any kind of central authority. Every time a transaction is validated, a new block is added to the chain and, typically, it cannot be removed afterwards [92].

The logic behind blockchain is inspiring for the construction of a robust parking system. Arguably, considering park consortiums, the first issue that has to be addressed is how anticompetitive behavior can be prevented. We also have to ensure that every member only acts with the aim of reducing traffic congestion. Blockchain mechanisms can be the answer to these problems. Introducing a consensus to validate any action from consortium members guarantees that no one can take advantage of the system. No member will be able to set a price which is too low with regards to the rest of the group. Even if they are not interested in reducing local traffic congestion, belonging to the network will enforce preservation of their economic interests. Such a mechanism, therefore, forces members to defend the entire process, while they naturally try to defend themselves. Members might disallow incorrect or dangerous behaviors from other components and, as a

consequence, they will cooperate to make the parking process work flawlessly.

The same holds for capacity variations. Especially during a pandemic, it is easy to imagine that anti-contagion measures impose restrictions on the number of slots that can be effectively occupied in a car park. Someone could try to be crafty and violate such restrictions, in order to have more customers than the concurrency, hence being able to gain larger profits. The consensus-based blockchain mechanism avoids a situation like that. The capacity of each car park is supposed to be known in advance, hence if someone tries to overcome the number of allowed slots, the network would recognize it and would stop it.

Essentially, blockchain logic is hereby used to avoid incorrect behaviors from network participants, therefore preserving the entire parking process, while keeping the reduction of traffic congestion as the main concern.

2.4.2 BC Parking Algorithm

Let us first describe the system representation through blockchain technology, by pointing out the reasons why one can took advantage from blockchain logic and what are the earned benefits. After that, it is possible to define some fairness constraints in order to guarantee a right competition, rules ensuring correct behavior of all the partici-

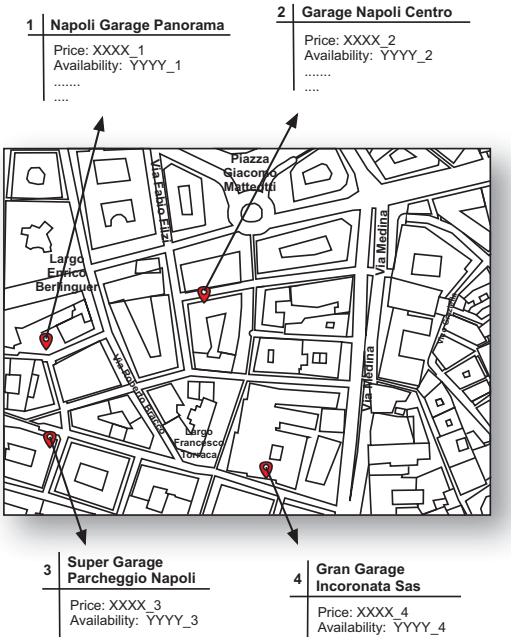


Figure 2.6: Example of parking areas which are consortium members

pants. Then, the pseudo-code of the proposed consensus algorithm is shown, which is needed to validate any modification operation on the blockchain. Finally, an overview of the benefits brought by such an approach from various points of view is provided.

Competitive-Blockchain-based Representation

We consider a realistic scenario made of several parking areas, which are members of a consortium but still work competitively, without affecting the free market laws. They will be called *competitors* in the sequel.

The introduction of such a competitive attitude, clearly, raises the problem of guaranteeing equity and fairness. When we move to a competitive setting, it is important to take into account that any par-

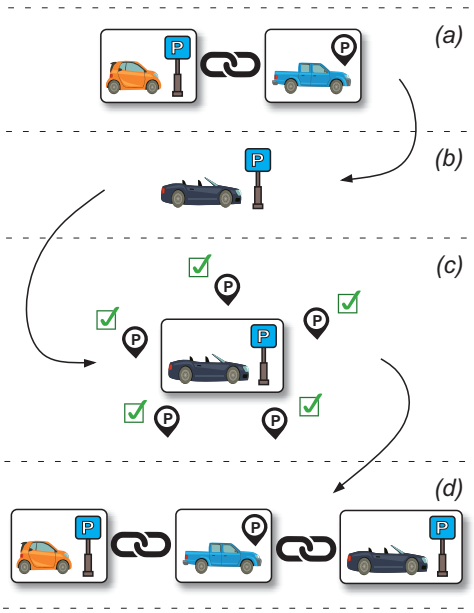


Figure 2.7: Schema of how the parking process works, by using the blockchain mechanism

ticipant wants to *win*, whatever it takes. It is quite easy to imagine that the single consortium-member could act to improve his personal gain, rather than choosing the strategy that reduces the traffic flow. Such an incorrect behavior has to be avoided at all costs and a proper protection mechanism is obtained by applying the logic behind the blockchain mechanism, introduced before, where any network modification must be approved, through a consensus, before it is applied.

Figure 2.6 shows a simple example of the system that is going to be defined. Let us point out that, in our proposal, we consider four different competitors in the center of Naples, characterized by the following attributes:

- the **price**, established by each competitors, is the rate-per-hour

required anytime a vehicle allocates a slot in the corresponding car park;

- the **capacity**, in terms of number of parking slots in the corresponding parking area: moreover, a percentage limit may be imposed, if we consider, for instance, restrictive measures as the ones for the COVID pandemic.

The competitors are the *nodes* of the blockchain, while the *transactions* are all the proposed actions which modify the system configuration and, therefore, they need a *consensus* from the network to be applied. A *block* of the blockchain is an association vehicle-slot, which has already been approved and hence is known by any participant.

In the described scenario, one can consider two possible transactions:

- One of the competitors receives a new vehicle from the *ready queue*. Such an operation needs a consensus, according to the considerations we made previously. It might be the case that a car park owner has reached the maximum capacity allowed for him, according to new anti-COVID restrictions (let's say 60% of the capacity), but he wants to cheat, and he keeps accepting more vehicles. Obviously, such an aggressive behavior has to be avoided, and other competitors will negate the consensus to accept new vehicles;

- One of the competitors changes his price. Again, a price variation should be acceptable, considering a fair competition. Let's say, for instance, that the average rate-per-hour of all the members of the consortium is 5$. Therefore we may have a member which applies a rate of 4$, one of 6$, but if someone wants to set a price of 1$ per hour, this would make the competition unfair. Most likely people would be encouraged to go to the cheapest car park, causing a traffic congestion. But actually, through the consensus mechanism, the other members will not allow such a price variation. They won't because they want to defend themselves, but consequentially they are also preserving the whole parking process.

Another possibility is that a vehicle leaves the corresponding parking slot, but in this case we assume that the consensus is not needed. The only focus is on the vehicle allocation process, hence a vehicle leaving the parking area essentially represents the end of such a process.

Figure 2.7 shows an example of the execution flow leading to the parking process:

1. At the beginning, the actual blockchain configuration is shown, meaning all the transaction that have already been accepted and are known by all the participants. In particular, it is commonly known that the orange vehicle is in the blue car park, and that

the blue vehicle is in the white car park (2.7.(a)).

2. A new transaction is submitted to the participants. It corresponds to the action of parking the black vehicle in the blue competitor (2.7.(b)).

3. The transaction is accepted according to the specific consensus algorithm used. In this example, any other competitor allows the transaction (2.7.(c)).

4. The transaction has been approved, a new block is added to the chain, and it becomes common knowledge that the black vehicle is in the blue car park (2.7.(d)).

Fairness Constraints

In order to prevent misbehaviors deriving from competitive environments and guarantee a fair competition between competitors, we define some *fairness constraints*.

Assumption Without loss of generality, it is assumed that, given the capacity $\mathcal{C}$ associated to some competitor, for reasons of force majeure, such as an anti-COVID measure, only a percentage of this capacity can be effectively occupied. Let us denote by c the actual capacity of each competitor.

One can define two distinct fairness constraints:

- A price fairness constraint, which guarantees the free market for any competitor, and informs the rest of consortium members of any price change, in order to keep the competition *fair*, precisely. For instance, if the competitor P_1 wants to raise the parking rate to a price which is considered too high for other competitors, such a transaction will be denied from the system, as well as, similarly to the example proposed in the previous section, if he wants to decrease the rate to a price considered too cheap, again the transaction will be denied.

- A capacity fairness constraint, which aims at guaranteeing that all the competitors follow the capacity restrictions shared by all the consortium members. For instance, if the competitor P_1 is trying to fill more parking slots than allowed, to increase his gain, such a transaction will be denied from the system.

Verification of the transactions must be on competitors side: establishing if a transaction does not violate the fairness constraints, and is hence authentic, is up to them. Essentially, these fairness constraints follow the democracy rules: if the majority of the participants accepts a given proposal, then it will be considered, otherwise it will not. The majority also guarantees fairness also in the preservation of fairness constraints. Obviously, one could negate the consensus just to limit the growth of the requiring member, while by introducing the majority we ensure an even judgment of the transactions.

2.4.3 The Consensus Algorithm

In literature, there are two main types of consensus algorithms [98]:

- proof of work-based ones (**PoW**, for short);
- pure stake-based ones (**PoS**, for short).

The main difference is that, with PoW, among all the nodes of the network, the one performing sufficient proof will get the right to append a new block, while in PoS the stake owned by each node determines if the node will modify the network (the basic idea is that the stake is proportional to the trust).

The consensus algorithm proposed for the solution, for the particular setting described before, has a different approach with respect to the just explained PoW and PoS consensus algorithms. It is made a simple, but crucial, consideration: each competitor would not attack the same chain which guarantees him a profit. The consensus algorithm, hence, is thought to guarantee the system's best interest, by letting each competitor do his own interest. In this case, the aim is an automatic parking system involving a consortium of parking areas which work in competitive fair manner. To guarantee fairness, one can rely on the competitors themselves. That is, respecting market laws and letting other competitors respect market laws is their best interest. As already mentioned, the traffic flow reduction is a mere consequence of members trying to improve their profits. And this is

the strength of the proposed model, because somehow no one is forced to do anything to reach the goal, they simply do it naturally.

For this reason, the consensus algorithm fitting the problem essentially requires the majority of approvals for a proposed transaction. The pseudo-code shown in Algorithm 2 defines the basic idea for the consensus.

Algorithm 2 Consensus Algorithm

```
1: yes = 0
2: for any competitor do
3:     if accepted transaction then
4:         yes++
5:     end if
6: end for
7: if yes ≥ #competitors/2 then
8:     add a new block to the chain
9: else
10:    leave the chain unchanged
11: end if
```

Essentially, the Algorithm 2 takes as input the proposed transaction that needs to be approved, and gives as output the updated blockchain: in particular, each competitor gives his opinion on the proposed transaction, voting on its approval. Once each competitor expresses his opinion, the votes are counted: if a majority has been reached, the transaction is approved and a new block is added to the chain, otherwise the same starting blockchain is returned.

Slot Assignment Process

Clearly, first it is needed to assign the vehicle to one of the competitors, then to a specific slot in it. Essentially, one has to specify how the car park is chosen for a given vehicle ready to be parked.

In order to do so, before applying the distance-based algorithm introduced before, a step is needed: the next vehicle in the queue is assigned to the most *similar* competitor, then the distance-based algorithm can be applied to select the best slot. What does *similar* mean? To clarify this concept, let us introduce only some of the additional characteristics for each competitor, collected in a feature vector:

- rate per hour;
- closing hours;
- key deposit;

In this way, any competitor is defined by the price per hour, the time by which the driver has to pick up his car, and the necessity to leave the key of the car. This last attribute is supposed to be binary (0 for no, 1 for yes).

One can assume that, similarly, any driver is characterized by the same 3-components-vector, which indicate his preferences on each feature. Let us also assume that the driver vector can have a special character $*$, which means *don't care*. This works like a *whatever* value. For instance, the vector $d = (*, *, 0)$ represents a driver who is only

interested in not leaving the key, while he doesn't care about the price and the closing time. In this case, the driver feature vector is not a single vector, but is represented by all the possible vector whose third component is zero.

The driver feature vector can be built in two different ways:

- explicitly: in this case the driver express his preferences;
- implicitly: in this case the system keeps track of the driver history and previous choices.

In such a setting, the competitor which is more suitable for a given driver is the one whose feature vector is closer (in terms of Euclidean distance) to the driver feature vector.

In order to compute a more precise similarity between driver and competitor, we sum up some of the most interesting features that could be taken into account to know the driver's preferences:

- Indoor/outdoor parking slots. Many people could prefer an indoor slot for their vehicle so the have it repaired from meteorologic inconveniences.
- Payment methods. Very often people are more comfortable paying by card, rather than by cash, especially in a pandemic era.
- Included shuttles toward center or airports. Very often, people need to leave their car before visiting a city or before reaching the airport, and hence they could be interested in such a facility.

- Included insurance against possible damages to the vehicle inside the car park.
- Included surveillance mechanism.
- Included gas station.
- Included charging points, in case of electric vehicles.
- Included car washing.
- Included Wi-Fi connection.

In case of *ambiguity*, meaning that the vector similarity does not produce a unique result, the choice falls on the competitor which is the closest to the destination of the driver.

The whole process is synthesized in Figure 2.8: when a new vehicle has to be parked, his feature vector is compared with the ones of the competitors; the most similar (meaning the closest in terms of Euclidean distance) is chosen, in this example, competitor p_3 is selected; inside the parking area associated to the competitor p_3, the slot which is the closest to the destination of the driver is chosen, according to the destination-based algorithm.

2.4.4 Model Validation

In order to further validate the behavior of the model, a minimal implementation has been developed using Rust, a system programming

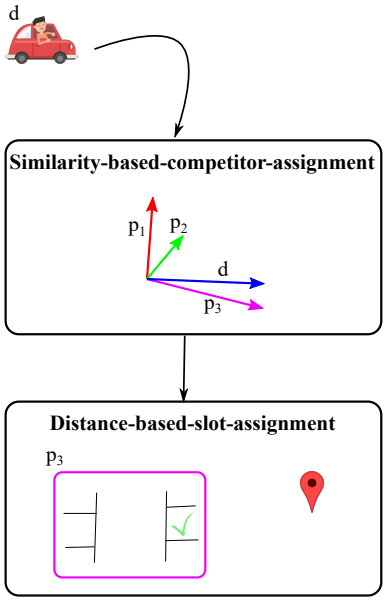


Figure 2.8: Steps of the parking process

language developed by Mozilla that, due to its particular memory management system, allows building fast, efficient and secure applications that can be executed on integrated devices using a small amount of resources.

It is also commonly used for blockchain applications development, due to the presence of the Substrate framework, which offloads the developers from the creation of the distributed network and automatically manages the state of the shared ledger.

Given a *consortium* (which holds a set of *parking*), and a *parking request* from a customer, the algorithm used to select the most appropriate parking lot is described in Algorithm 3.

If a valid parking lot is found, the blockchain receives a new proposed block, which has to be accepted by the majority of the competitors, as already explained in Algorithm 2. Once the consortium

Algorithm 3 Parking lot selection

```
candidates ← [ ]                                   ▷ Filters parking lots
for parkinglot ∈ consortium.parking_lots do
    if parkinglot.distance(request.location) < 5km then
        if parkinglot.requested_services(request) then
            candidates.append(parkinglot)
        end if
    end if
end for          ▷ Sort by optional offered services, then by distance
candidates.sort(optional_services, distance) return candidates.first()
```

accepts the block it becomes part of the blockchain, in order to keep the process transparent for all the participants.

Notice that the constraint of a distance within *5km* to find a parking slot should be relaxed if there isn't a slot satisfying such a property.

Table 2.9 shows five different simulated results with 10 parking lots and 1000 requests, and shows that the system is actually able to assign users evenly to the various competitors.

Table 2.9: Simulation: 10 parking lots/1.000 requests, numbers indicate how many vehicles have been allocated

Lot	R1	R2	R3	R4	R5
Parking A	83	66	95	114	126
Parking B	72	40	102	106	79
Parking C	70	78	92	101	69
Parking D	93	110	130	67	66
Parking E	110	94	178	56	174
Parking F	58	73	58	123	168
Parking G	122	177	85	122	64
Parking H	113	42	70	61	126
Parking I	139	164	133	124	41
Parking J	140	156	57	126	87

2.4.5 Benefits

It is interesting to point out which are the advantages brought by the system proposed in this work, as shown in Figure 2.9.

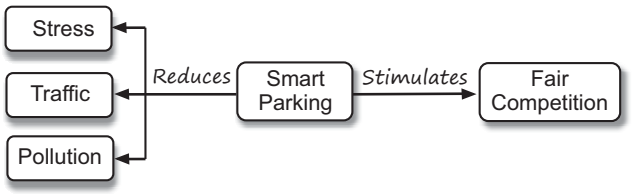


Figure 2.9: Advantages of the proposed system

Let us analyze different points of view:

- Driver point of view: according to the distance based algorithm proposed in [19], which is still used in this work, each driver will place his car in the parking slot which is as close as possible to his final destination. Such a criterion guarantees low time wasting to each the target (for instance a clothes shop) from the slot where the vehicle has been left. Moreover, the structure of the algorithm itself ensures that the time to empty the *ready* queue is minimized, and as a consequence also the waiting time for each driver is reduced.

- Competitors point of view: the fairness constraints handled through a blockchain mechanism ensure a fair competition between the members of the same consortium, by not affecting market rules and guaranteeing correctness. Each competitor is somehow pro-

tected by the consortium itself, and a fair profit is guaranteed to any of them.

- Consortium point of view: while protecting themselves, competitors also protect the whole consortium, acting in its best interest, even if they have the feeling of acting for their benefit only.

- Environment point of view: reducing the time typically wasted while searching for a free parking slot dramatically decreases the amount of gas emission, contributing to reduction of air pollution.

- Traffic point of view: as the previous case, traffic congestion is typically affected by drivers slowing down or cruising along the same streets over and over. Minimizing the time to complete the parking process, traffic conditions will also improve.

Part III

Part Two

Chapter 3

Reasoning About DEL games

3.1 Introduction

Many applications fall within the scope of reachability games with imperfect information (economics, robotics, distributed computing systems, web services, etc), such as video games [46] (Civilization, etc.), Kriegspiel (the epistemic variant of Chess) [86], Hanabi [16], or contingent and conformant planning [59]. For instance, drones patrolling an area may have to decide which trajectory to take so that the status (safe or unsafe) of each zone in this area is always known to at least one of them, while antagonistic agents try to keep the status of some areas secret.

Games with imperfect information are computationally hard, and

even undecidable for multiple players [107]. One way to tame this complexity is to make assumptions on how the knowledge of the different players compare: if all players that cooperate can be ordered in a hierarchy where one knows more than the next, a situation called *hierarchical information*, then the existence of distributed strategies can be decided [32, 105]. Another natural approach is to consider fragments based on classes of action types, as done for instance in [28, 38, 112] where different kinds of public actions are considered.

By contrast, *Dynamic epistemic logic* (DEL) [129] was designed to describe actions precisely: how they affect the world and how they are perceived by agents. In particular, classic action types such as *public/private announcements* or *public actions* correspond to natural classes of DEL action models. Also, DEL extends epistemic logic and hence enables modelling higher-order knowledge, i.e., what an agent knows about what another agent knows etc, and the evolution of agents' knowledge over time.

This work bridges the gap between DEL and games by introducing adversarial aspects in DEL planning, thus moving from plan generation to *strategy synthesis*. Two frameworks are defined for DEL-based reachability games, where players start in a given epistemic situation and their possible moves are described by action models, and the objective is to reach a situation satisfing some epistemic formula.

First, open systems are considered [63], i.e., systems that interact

with an environment. In this setting (called *DEL controller synthesis*), two omniscient, external entities (controller and environment) choose in turn which actions are performed, while agents involved in the models and formulas are not active, they merely observe how the system evolves based on the actions chosen by the controller and the environment, and update their knowledge accordingly.

DEL controller synthesis extends DEL planning, as the latter is a degenerate case of the former where the environment stays idle, and therefore inherits undecidability for the general case.

Precisely, it will be shown PSPACE-completeness when possible moves are public announcements, EXPTIME-completeness for the more general public actions, and membership in $(k+1) - \text{EXPTIME}$ for propositional actions when the objectives are formulas of modal depth at most k.

Second, a further generalization is applied, by turing agents into players: unlike the omniscient controller of the previous setting, agents have imperfect information about the current state of the game, and can only base their decisions on what they know. This is modeled through *uniform strategies*. In this setting, one can study the *distributed strategy synthesis* where a group of players cooperate to enforce some objective against the remaining players. The problem is undecidable, already for propositional actions and a coalition of two players. However, two kinds of assumptions that make imperfect-

information games decidable, namely public actions and hierarchical information, also yield decidable cases of multiplayer DEL games, with a complexity which is not worse than the case of controller synthesis for public actions and public announcements.

Typically, situations like the one described above are handled through imperfect information game arenas, i.e., graphs whose nodes represent positions of the game, edges are the possible actions, and equivalence relations capture indistinguishability of positions. However, the state explosion problem makes game structures often very large, making distributed synthesis intractable. In order to circumvent this difficulty, implicit descriptions by means of *DEL presentations* can be employed instead of explicit representations of game structures. A DEL presentation is made of a finite initial epistemic model that reflects the initial knowledge of the agents, and a finite set of epistemic actions available to them and the other agents in the environment. This setting is very convenient to define various types of actions such as public actions or semi-private announcements, and study how restricting to such actions can make distributed synthesis easier.

Precisely, agents are split into two antagonistic teams, $Agt_\forall$ and $Agt_\exists$, such that agents in $Agt_\exists$ want to reach some goal while the ones in $Agt_\forall$ try to prevent them from winning. Also in this setting, decidability can be obtained in the case of public actions and hierarchical information. In particular, reachability goals are defined through win-

ning conditions expressed in LTLK, that blends temporal operators and epistemic modalities.

To prove decidability results, a crucial step is to show that the game arena induced by a DEL game presentation, which is in general infinite, can be represented finitely. This was already known for the case of actions whose preconditions do not involve knowledge but are purely propositional formulas [53, 89], and it is used to transfer an existing result for distributed synthesis in explicit game arenas with hierarchical information.

It is also proved that the infinite game generated from a DEL game presentation is regular also in the case of public actions. This is done by observing that, modulo isomorphism, such actions can only generate finitely many different epistemic models from the initial one, thus allowing us to get an equivalent finite game as the quotient of the infinite one. Moreover, for public announcements and the syntactic fragment of LTLK, without next operator and local knowledge properties only, one can show an even stronger characteristic of game arenas that allows us reducing to a polynomial-length horizon game and to derive an optimal PSPACE procedure.

Finally, concurrent executions of actions are considered in epistemic planning, which are essential for modelling realistic situations, providing two main contributions.

First, defining a *DEL concurrent update product* that provides the

dynamics of concurrent epistemic actions. This new product yields a non-deterministic dynamics controlled by a scheduler, whose role is to resolve conflicts. On the basis of this product, it is possible to show how finite DEL presentations can generate infinite concurrent game arenas.

Second, a proof that distributed synthesis can be solved for such games for two cases: the case where all actions are public (perfectly observed by everyone), and the case where all actions have propositional pre- and post-conditions and information among the agents is hierarchical. This is proved by showing that the infinite concurrent game arenas arising from DEL game presentations can be finitely represented when actions are all public or all propositional. In this way, one can transfer existing results on the model-checking problem for the epistemic strategic logic $\mathsf{ATL}^*_{\mathrm{K}}$ [127] to obtain, in particular, decidability of distributed synthesis for temporal epistemic objectives.

3.2 Related Work

The complexity of DEL-based epistemic planning has been thoroughly investigated. It is undecidable already for actions with preconditions of modal depth one and propositional postconditions [35, 73]. For preconditions of modal depth one and no postconditions the problem has been open for years, but it is decidable when pre- and postconditions are propositional [15, 53, 139]. It is also known to be NP-complete for

public announcements [36, 44], and Pspace-complete for public actions [44]. The decidability for propositional actions has been extended in [15] by considering infinite trees of actions called protocols instead of finite plans, and specifications in branching-time epistemic temporal logic instead of reachability for epistemic formulas; this has been extended further in [53] by enriching the specification language with Chain Monadic Second-order Logic. Both results rely on the fact that when actions are propositional, the infinite structures generated by repeated application of action models form a class of regular structures [15, 87], i.e., relational structures that have a finite representation via automata. First-order logic is decidable on such structures [33], and chain-MSO is decidable on a subclass called regular automatic trees [53], but neither of these logics can express the existence of strategies in games. Here it will be shown that the regular structures obtained from propositional DEL models can be seen as finite turn-based game arenas studied in games played on graphs and, hence, decidability results on games with epistemic temporal objectives can be transferred to the DEL setting [91].

Regarding concurrency aspects introduced in DEL, concurrent execution of actions has not been considered so far in epistemic planning. Some works consider concurrent execution of abstract actions (as in concurrent game structures [48]), or concurrent execution of purely epistemic actions without effects on the world [4, 30, 130] or only pub-

lic [49]. But concurrent execution of arbitrary epistemic actions with explicit effects has never been studied. And yet they are essential for modelling realistic situations.

3.3 Background on Games and Epistemic Planning

3.3.1 Notations

Let us some standard notions and notations that will be needed in the rest of the work. A *finite* (resp. *infinite*) *word* over some alphabet Σ is an element of Σ^* (resp. Σ^ω). The *length* of a finite word $w = w_0 w_1 \dots w_n$ is $|w| := n+1$, and $last(w) := w_n$ is its last letter. Given a finite (resp. infinite) word w and $0 \leq i < |w|$ (resp. $i \in \mathbb{N}$), we let w_i be the letter at position i in w, $w_{\leq i} := w_0 \dots w_i$ is the prefix of w that ends at position i and $w_{\geq i} := w_i w_{i+1} \dots$ is the suffix of w that starts at position i.

3.3.2 Game arenas

Along this work different types of game arenas will be considered, depending on the number of players, whether they play in turn or concurrently, whether they observe perfectly the situation or not, and finally whether transitions are deterministic or non deterministic. Be-

cause many concepts such as plays, histories or strategies are common to all cases, let us first introduce the most general case of multiplayer concurrent arenas with non deterministic transitions and imperfect information. Then the other classes of arenas will be defined as particular cases.

General definition

Let us fix a finite set of *atomic propositions* AP as well as a finite set of *agents* $Agt = \{a_1, \ldots, a_N\}$.

Definition 3.3.1. *An* arena *is a tuple* $G = (Act, V, V_\iota, RAct, \Delta, \preccurlyeq, \lambda)$*, where*

- Act *is a non-empty set of* actions*,*
- V *is a non-empty set of* positions*,*
- $V_\iota \subseteq V$ *is a non-empty set of* initial positions*,*
- $RAct : Agt \to V \to 2^{Act} \setminus \{\emptyset\}$ *maps each agent and position to a non-empty repertoire of actions available to this agent in this position,*
- $\Delta \subseteq V \times Act^N \times V$ *is a* transition relation*,*
- $\preccurlyeq$: $Agt \to 2^{V \times V}$ *maps each agent a to an* epistemic relation *over positions, and*
- $\lambda : V \to 2^{AP}$ *is a* valuation function*.*

For convenience we will write $Act_a(v)$ for $RAct(a)(v)$ and $\preccurlyeq_a$ for $\preccurlyeq(a)$.

In a position v each agent a chooses an available action $\alpha_a \in Act_a(v)$, yielding a *joint action* $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_N \rangle$. We write $JAct(v)$ for the set of joint actions available in v. The game then moves nondeterministically to some position $v' \in \Delta(v, \vec{\alpha})$. We call *play* an infinite sequence of positions $\pi = v_0 v_1 v_2 \ldots$ such that for all $k \in \mathbb{N}$, there exists $\vec{\alpha} \in Act^N$ such that $v_{k+1} \in \Delta(v_k, \vec{\alpha})$. A *history* is a finite nonempty prefix of a play. We write $Plays^G$ and $Hist^G$ for the sets of plays and histories in G, respectively (the superscript might be omitted when the game is clear from the context). One may also write $\lambda(h)$ for $\lambda(last(h))$, meaning the set of atomic propositions that hold in the last position of history h. Similarly one can write $Act_a(h)$ for $Act_a(last(h))$.

Epistemic relations

The epistemic relation $\preccurlyeq_a$ indicates how agent a perceives positions: $v \preccurlyeq_a v'$ means that when in position v, agent a considers it possible that v' is the actual position. These relations will often be equivalence relations, in particular when they are associated with agents that must act based on their knowledge, as usual in games with imperfect information [12]. This will be made clear by writing $\approx_a$ instead of $\preccurlyeq_a$, and in that case one can say that agent a has *S5 knowledge* [56].

Games generated from DEL models will be studied, which naturally model agents who have perfect recall of the past and have access to a global clock counting the number of events (although other interpretations are possible, see [50]). As a result the classic *synchronous perfect recall* extension of indistinguishability relations to histories will be used (see for instance [56] for more on perfect recall). The indistinguishability relation of each agent a is lifted to histories as follows: $h \preccurlyeq_a h'$ if $|h| = |h'|$ and $h_i \preccurlyeq_a h'_i$ for every $i < |h|$.

Playing with imperfect information

Intuitively, in games with imperfect information, players' strategies must assign the same actions to indistinguishable situations [100]. For this to make sense, it is necessary to assume that players have S5 knowledge, i.e., their epistemic relations are equivalence relations[1]. However in some of the considered games, some agents do not actively play but passively observe, and for such agents it is not needed to make this restriction. An agent is an *observer* if $Act_a(v)$ is a singleton for every position $v \in V$, so that it never has any real choice to make. Otherwise agent a is a *player*. For clarity, the special action *idle* will be used whenever an agent is not playing. From now on, it is assumed that *players* have S5 knowledge, while *observers* can have arbitrary epistemic relations.

[1] Actually the usual definition seems to make sense also for KD45 knowledge, i.e., for relations that are serial, transitive, and Euclidean. But this would be highly non-standard, and we prefer to stick to usual S5 knowledge.

A *strategy* σ for an agent a is a function $\sigma_a : Hist^G \to Act$ such that for every history h, $\sigma_a(h) \in Act_a(h)$. Note that by definition, an observer has only one possible strategy, which is to always play *idle*. If an agent a is a player, it is required in addition that its strategies are *uniform*, meaning that for every pair of indistinguishable histories $h \approx_a h'$, $\sigma_a(h) = \sigma_a(h')$. Classically, it is assumed that $Act_a(v) = Act_a(v')$ whenever $v \approx_a v'$ (if available actions were different, the player could tell the difference between the positions).

In addition, if the game is turn-based (see Section 3.3.2 below), it is required that players always know whose turn it is to play: for every player a, if $v \approx_a v'$ then there exists a player b such that $v, v' \in V_b$.

Classes of game arenas

If the epistemic relation $\preccurlyeq_a$ of each player is the identity relation, then G is a game arena with *perfect information*.

If, in every position v, there is exactly one agent a such that $Act_a(v) \neq \{idle\}$, then we say that the game is *turn-based*. In that case, one can partition the set of positions as $V = \uplus_{a \in Agt} V_a$, where $V_a = \{v \mid Act_a(v) \neq \{idle\}\}$.

If, for every position v and joint action $\vec{\alpha}$, $\{v' \mid (v, \vec{\alpha}, v') \in \Delta\}$ is a singleton, then G is *deterministic* and one may represent the transition relation as a function $\delta : V \times Act^N \to V$.

Winning strategies

A *strategy profile* for a coalition of agents $C \subseteq Agt$ is a tuple $\sigma_C = \langle \sigma_a \rangle_{a \in C}$, and Σ_C is the set of strategy profiles for a coalition $C \subseteq Agt$. An *outcome* of a strategy profile σ_C from a position v_0 is a play starting in position v_0 and in which agents in C follow the strategies in Σ: formally, a play $\pi = v_0 v_1 \dots$ is an outcome of $\langle \sigma_a \rangle_{a \in C}$ from v_0 if, for every $k \in \mathbb{N}$, there is a joint action $\vec{\alpha}$ such that $v_{k+1} \in \Delta(v_k, \vec{\alpha})$ and $(\vec{\alpha})_a = \sigma_a(v_0 \dots v_k)$, for all $a \in C$. We let $Out(v, \sigma_C)$ be the set of outcomes of σ_C from position v, and for a set $V' \subseteq V$ of positions we let $Out(V', \sigma_C) = \cup_{v \in V'} Out(v, \sigma_C)$.

A *winning condition* for a game arena G is a subset of plays $Win \subseteq Plays^G$. Given a game arena G, a coalition $C \subseteq Agt$ and a winning condition Win for C, a strategy profile σ_C for C is *winning* if all its outcomes from starting positions are winning, i.e., $Out(V_\iota, \sigma_C) \subseteq Win$.

Game unfolding

Finally, let us introduce the unfolding of a game arena, which will be used to define an equivalence relation on game arenas.

Definition 3.3.2. *Given a game arena* $G = (Act, V, V_\iota, RAct, \Delta, \preccurlyeq, \lambda)$, *the* unfolding *of* G, $G^{\text{unf}} = (Act, V', V'_\iota, RAct', \Delta', \preccurlyeq', \lambda')$, *is the game arena where*

- $V' = Hist^G$;

- $V'_\iota = V_\iota$;
- *for every* $h \in Hist^G$ *and every agent* $a \in Agt$, $Act'_a(h) = Act_a(last(h))$;
- $\Delta' = \{(h, \vec{\alpha}, h \cdot v) \mid (last(h), \vec{\alpha}, v) \in \Delta\}$;
- $\preccurlyeq'_a$ *is the synchronous perfect-recall lifting of* $\preccurlyeq_a$ *to histories;*
- $\lambda'(h) = \lambda(last(h))$.

One can say that two game structures G and G' are *equivalent* whenever their unfoldings are isomorphic.

3.3.3 The classic DEL setting

Le us now recall the models used in Dynamic Epistemic Logic to describe epistemic situations and epistemic actions, as well as the update product that captures their dynamics.

Epistemic models

Let us recall models of epistemic logic [56].

Definition 3.3.3. *An* epistemic model $\mathcal{M} = (W, (\preccurlyeq_a)_{a \in Agt}, \lambda)$ *is a tuple where*

- W *is a set of* worlds *(or situations),*
- $\preccurlyeq_a \subseteq W \times W$ *is an* accessibility relation *for agent* a*, and*

- $\lambda : W \to 2^{AP}$ *is a* valuation function.

One ca write $w \preccurlyeq_a u$ instead of $(w, u) \in \preccurlyeq_a$; the intended meaning of $w \preccurlyeq_a u$ is that when the actual world is w, agent a considers that u may be the actual world. The valuation function λ provides the subset of atomic propositions that hold in a world. A pair $(\mathcal{M}, w)$ where w is a world in $\mathcal{M}$ is called a *pointed epistemic model*, or *epistemic state*, while a pair $(\mathcal{M}, W')$, where $W' \subseteq W$ is a subset of worlds, is called a *multipointed epistemic model*.

An epistemic model is *finite* if its set of worlds W is finite and for each world $w \in W$, $\lambda(w)$ is finite. In that case, we let $|\mathcal{M}|$ be the *size* of $\mathcal{M}$, defined as $|W| + \sum_{a \in Agt} |\preccurlyeq_a| + \sum_{w \in W} |\lambda(w)|$. From now on, all epistemic models are assumed to be finite.

Epistemic Logic

The syntax of Epistemic Logic **EL** is given by the following grammar:

$$\phi \ ::= \ p \ \mid \ \neg\phi \ \mid \ \phi \vee \phi \ \mid \ K_a\phi$$

where p ranges over AP and a ranges over Agt.

$K_a\phi$ is read 'agent a knows that ϕ is true'. We define the usual abbreviations $\top$ for $p \vee \neg p$, $\bot$ for $\neg\top$, $\phi_1 \wedge \phi_2$ for $\neg(\neg\phi_1 \vee \neg\phi_2)$ and $\hat{K}_a\phi$ for $\neg K_a \neg\phi$, and use Prop for propositional formulas, the fragment of **EL** obtained by removing the knowledge operator. The *modal depth* of a formula is its maximal number of nested knowledge operators; for

instance, the formula $K_a K_b p \wedge \neg K_a q$ has modal depth 2. The *size* $|\phi|$ of a formula ϕ is the number of symbols in it.

Formulas of **EL** are evaluated on pointed epistemic models.

Definition 3.3.4. *Let us define $\mathcal{M}, w \models \phi$, read as 'formula ϕ holds in the pointed epistemic model $(\mathcal{M}, w)$', by induction on ϕ, as follows:*

- $\mathcal{M}, w \models p$ *if* $p \in \lambda(w)$;
- $\mathcal{M}, w \models \neg\phi$ *if it is not the case that* $\mathcal{M}, w \models \phi$;
- $\mathcal{M}, w \models \phi \vee \psi$ *if* $\mathcal{M}, w \models \phi$ *or* $\mathcal{M}, w \models \psi$;
- $\mathcal{M}, w \models K_a\phi$ *if for all* u *such that* $w \preccurlyeq_a u$, $\mathcal{M}, u \models \phi$.

Sometimes, variables x that range over some finite domain will be used. In such cases, one may write $(x = d)$ to express that "the value of x is d". This can be encoded with atomic propositions x_d, which hold true when x is equal to d and false otherwise.

Action models

In addition to epistemic models, Dynamic Epistemic Logic (DEL) also relies on *action models* (also called "event models"). These models specify actions, the preconditions for their execution, their effects on the world, and how agents perceive their occurrence.

Definition 3.3.5. *An* action model $\mathcal{A} = (A, (\preccurlyeq_a^{\mathcal{A}})_{a \in Agt}, pre, post)$ *is a tuple where:*

- A *is a set of possible* actions,
- $\preccurlyeq_a^{\mathcal{A}} \subseteq A \times A$ *is the* accessibility relation *for agent* a,
- $pre : A \to \mathbf{EL}$ *is a* precondition *function, and*
- $post : A \times AP \to Prop$ *is a* postcondition *function.*

An action α is *executable* in a world w of an epistemic model $\mathcal{M}$ if its precondition $pre(\alpha)$ holds in w, i.e., $\mathcal{M}, w \models pre(\alpha)$. We assume that all actions' preconditions are satisfiable. A set of actions $A' \subseteq A$ is *non-blocking* if $\bigvee_{\alpha \in A'} pre(\alpha) \equiv \top$, i.e., there is always at least one action in A' that is executable. After executing an executable action α in a world w, proposition p holds if its postcondition was satisfied before executing the action; thus, let us define $\lambda(w, \alpha) := \{p \in AP \mid \mathcal{M}, w \models post(\alpha, p)\}$ as the set of propositions holding after executing α in w. When postconditions are propositional, one can define similarly $\lambda(\nu, \alpha)$ where $\nu \subseteq 2^{AP}$ is a valuation. A *pointed action model* is a pair $(\mathcal{A}, \alpha)$ where α represents the actual action.

Only finite action models will be considered, i.e., such that the set of actions A is finite, and for every action $\alpha \in A$ there are only finitely many atomic propositions $p \in AP$ whose postcondition is not trivially false, i.e., such that $post(\alpha, p) \neq \bot$. We let $|\mathcal{A}|$ be the *size* of $\mathcal{A}$, defined as follows:

$$|\mathcal{A}| := |A| + \sum_{a \in Agt} | \preccurlyeq_a^{\mathcal{A}} | + \sum_{\alpha \in A} |pre(\alpha)| + \sum_{\alpha \in A, p \in AP} |post(\alpha, p)|$$

When working with variables x over finite domains, one may write $x := d$ for the effect of setting x to value d. This can again be encoded with atomic propositions x_d and postconditions as defined above.

Update product

After occurrence of an action α in a world w, agent a considers it possible that action α' occurred in world w', if in w he considers w' possible and α' is executable in w'. Hence, he considers action α' possible when action α is executed. This leads to the following definition of the product that models how to update an epistemic model when an action is executed [18].

Definition 3.3.6 (Product [18])**.** *Let* $\mathcal{M} = (W, (\preccurlyeq)_{a \in Agt}, \lambda)$ *be an epistemic model, and* $\mathcal{A} = (A, (\preccurlyeq_a^{\mathcal{A}})_{a \in Agt}, pre, post)$ *be an action model. The* product *of* $\mathcal{M}$ *and* $\mathcal{A}$ *is defined as* $\mathcal{M} \otimes \mathcal{A} = (W', (\preccurlyeq_a^{\mathcal{A}})', \lambda')$ *where:*

- $W' = \{(w, \alpha) \in W \times A \mid \mathcal{M}, w \models pre(\alpha)\}$,
- $(w, \alpha) \preccurlyeq'_a (w', \alpha')$ *if* $w \preccurlyeq_a w'$ *and* $\alpha \preccurlyeq_a^{\mathcal{A}} \alpha'$, *and*
- $\lambda'(w, \alpha) = \lambda(w, \alpha)$.

Example Figure 3.1 shows the pointed model $\mathcal{M}, w$ that represents a situation in which p is true and both agents a and b do not know it. The pointed action model $\mathcal{A}, \alpha$ describes the action where agent

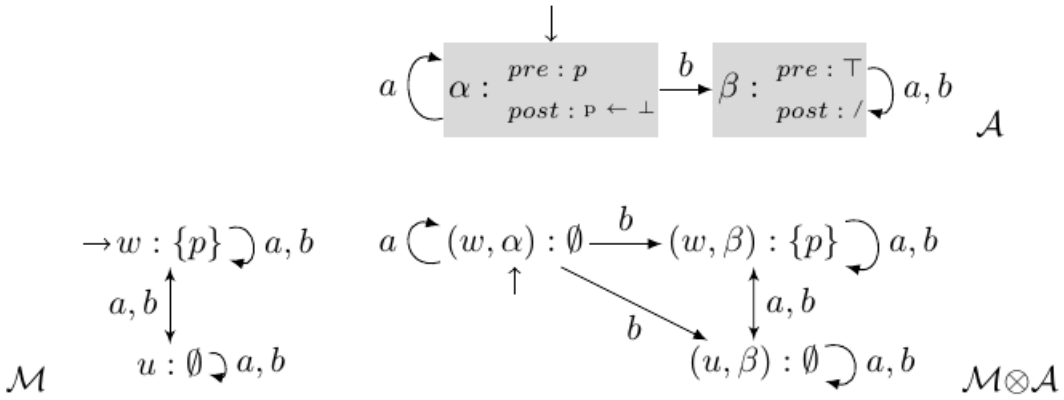


Figure 3.1: Example of DEL product. Symbol / indicates the trivial postcondition that leaves valuations unchanged.

a learns that p was true but that it is now set to false, while agent b does not learn anything (he sees action β that has trivial pre- and postcondition). In the product epistemic model $(\mathcal{M} \otimes \mathcal{A}, (w, \alpha))$, agent a now knows that p is false, while b still does not know the truth value of p, or whether agent a knows it.

An epistemic or action model is *S5* if all accessibility relations are equivalence relations. This property is important to model games with imperfect information, and it will be assumed in the sequel.

3.3.4 Generated structure

Iteratively executing an action model from an initial epistemic model generates an infinite sequence of epistemic models, whose union yields an infinite epistemic structure where dynamics are represented by the possible sequences of actions, while information is captured by the accessibility relations.

Definition 3.3.7 (Generated structure)**.** *Given an epistemic model $\mathcal{M}$ and an action model $\mathcal{A}$, let us define the family of disjoint epistemic models $\{\mathcal{MA}^n\}_{n\geq 0}$ by letting*

$$\mathcal{MA}^0 = \mathcal{M} \quad \text{and}$$
$$\mathcal{MA}^{n+1} = \mathcal{MA}^n \otimes \mathcal{A}$$

Le us finally define the infinite epistemic model $\mathcal{MA}^* = \bigcup_{n\in\mathbb{N}} \mathcal{MA}^n$.

One can identify objects of the form $(\dots((w, \alpha_1), \alpha_2), \dots \alpha_n)$ with $(w, \alpha_1, \dots, \alpha_n)$, that one may also write $w\alpha_1 \dots \alpha_n$. For every world $(w, \alpha_1, \dots, \alpha_n) \in \mathcal{M} \otimes \mathcal{A}^n$, and every formula $\phi \in \mathbf{EL}$, one may thus write $\mathcal{MA}^*, w\alpha_1 \dots \alpha_n \models \phi$ for $\mathcal{M} \otimes \mathcal{A}^n, (w, \alpha_1, \dots, \alpha_n) \models \phi$. This shows that the alternative definition of epistemic planning given in the next section is equivalent to the usual one.

3.3.5 Epistemic planning

In epistemic planning, the plan existence problem asks for the existence of an executable sequence of actions $\alpha_1, \dots, \alpha_n$ in an action model $\mathcal{A}$, whose execution from $(\mathcal{M}, w_\iota)$ leads to a situation satisfying some objective expressed as an epistemic logic formula. Formally, let us consider the following problem.

Definition 3.3.8 (Plan existence problem)**.**

- *Input:*

	Public announcements	Public actions	Propositional actions	Full
Plan	NP-c	PSPACE-c	decidable	undecidable
Controller	PSPACE-c (Th. 3.4.2)	EXPTIME-c (Th. 3.4.3)	decidable (Th. 3.4.4)	undecidable
Distributed strategy	PSPACE-c (Th. 3.5.2)	EXPTIME-c (Th. 3.5.3)	undecidable (Th. 3.5.1) decidable case (Th. 3.5.4)	undecidable

Table 3.1: Known results (new in grey) for plan, controller and distributed strategy synthesis.

  - *a pointed epistemic model* $(\mathcal{M}, w_\iota)$ *(the initial situation)*
  - *an action model* $\mathcal{A}$ *(possible actions)*
  - *a formula* $\phi \in \mathbf{EL}$ *(the objective)*

- *Output: yes if there exists* $\alpha_1 \dots \alpha_n$ *such that* $\mathcal{M}\mathcal{A}^*, w_\iota\alpha_1 \dots \alpha_n \models \phi$.

Remark 1. *Note that usual formulations of the plan existence problem consider a set of distinct pointed action models* $(\mathcal{A}_1, \alpha_1), \dots, (\mathcal{A}_n, \alpha_n)$ *instead of one action model* $\mathcal{A}$*. Both formulations are equivalent, in the sense that they are interreducible in linear time. One direction is clear; for the other, one can define* $\mathcal{A}$ *as the disjoint union of the* $\mathcal{A}_i$ *for all* i*, and add special atomic propositions that become true when an action is performed so that the goal formula can be used to ensure that only actions among* $\{\alpha_1, \dots, \alpha_n\}$ *are executed.*

Main known results on the plan existence problem are summarised in Table 3.1.

3.4 Controller Synthesis

The plan existence problem does not consider the possibility that events from the environment could come in the way of the correct execution of the plan. The first generalization consists in considering such an antagonistic environment, and asking for a plan that would ensure to reach the objective independently of how the environment behaves. This amounts to considering a two-player game between players that are called *Controller* and *Environment*, and looking for a winning strategy for *Controller*. Similarly to DEL epistemic planning, where the plan is chosen by some external entity that is none of the agents in the models, here *Controller* and *Environment* are two players distinct from the agents and, unlike the agents, they have perfect observation of the situation, so that we are dealing with two-player games of perfect information. The agents in the epistemic and action models are observers, whose knowledge is only used to evaluate whether the goal epistemic property holds.

3.4.1 The controller synthesis problem

Let us formally consider an initial epistemic model $\mathcal{M}$, defined as in Definition 3.3.3, with an initial world w_ι, and an action model $\mathcal{A} = (A, (\preccurlyeq_a^{\mathcal{A}})_{a \in Agt}, pre, post)$ whose set of actions A is partitioned into a set of actions A_{ctr} controlled by *Controller*, and a set of actions A_{env} controlled by *Environment*. In order to avoid deadlocks, it is required

that there always be at least one available action for both *Controller* and *Environment* (an easy way to achieve this is to include the *idle* action, which is perfectly observed by all, having true precondition and no effect on atomic propositions).

Definition 3.4.1. *A* DEL Controller-Environment-arena*, or* DEL C-E-arena*, is a triple* $\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle$ *where* $(\mathcal{M}, W_\iota)$ *is a multipointed epistemic model, and* $\mathcal{A}$ *is an action model with sets of actions* $A = A_{ctr} \uplus A_{env}$*, where both* A_{ctr} *and* A_{env} *are non-blocking.*

Given a DEL C-E-arena $\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle$ we define a turn-based game arena $G^{\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle}$ over agents $Agt \uplus \{\mathrm{Con}, \mathrm{Env}\}$ where Con and Env are players with perfect information, and agents in Agt are observers.

Definition 3.4.2. *Given a DEL C-E-arena* $\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle$*, one can formally define the deterministic turn-based arena* $G^{\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle}$ *as the tuple* $(Act, V, V_\iota, RAct, \delta, \preccurlyeq, \lambda)$*, where* $A = A_{ctr} \uplus A_{env}$ *for the set of actions in* $\mathcal{A}$ *and* $\mathcal{MA}^n = (W^n, (\preccurlyeq_a^n)_{a \in Agt}, \lambda^n)$ *for every* n *and:*

- *the set of actions is* $Act = A \uplus \{idle\}$*;*
- *the set of positions is* $V = \cup_{n \in \mathbb{N}} W^n$*, partitioned as follows:* $V_{Env} = \cup_n W^{2n}$*,* $V_{Con} = \cup_n W^{2n+1}$*, and* $V_a = \emptyset$ *for all other agents* a*;*
- *the set of initial positions is* $V_\iota = W_\iota$*,*

- *the agents' repertoires of actions are defined as follows:*

$$Act_{Env}(v) = \{\alpha \in A_{env} \mid v \models pre(\alpha)\} \qquad \text{if } v \in V_{Env}$$
$$Act_{Con}(v) = \{\alpha \in A_{ctr} \mid v \models pre(\alpha)\} \qquad \text{if } v \in V_{Con}$$
$$Act_a(v) = \{idle\} \qquad \text{in all other cases;}$$

- $\delta(v, \vec{\alpha}) = v \cdot \alpha$, *where* α *is the only non-idle action in* $\vec{\alpha}$*;*

- *agents' epistemic relations are as follows:*

  - $\preccurlyeq_{Con}$ *and* $\preccurlyeq_{Env}$ *are the identity relation*

  - *for each other agent* a, $\preccurlyeq_a = \cup_n \preccurlyeq_a^n$*;*

- $\lambda(v) = \lambda^n(v)$, *where* n *is such that* $v \in W^n$.

Controller and Environment play in turn: in each round, Environment first chooses to execute an action in A_{env}, then Controller reacts with some action in A_{ctr}. Other agents merely observe the evolution of the system. The problem of interest is then to decide whether Controller has a strategy to ensure that a situation satisfying some epistemic property, expressed by some formula $\phi \in \mathbf{EL}$, will eventually be reached. Given a game G, this objective is captured by the following winning condition:

$$Win_\phi^G = \{\pi \in Plays^G \mid \exists n, \pi_n \models \phi\}$$

One can use the fact that equivalence of game arenas (i.e., isomorphy of their unfoldings, see Section 3.3.2) preserves winning strategies for this winning condition.

Lemma 3.4.1. *Let G and G' be two equivalent game arenas, and let $\phi \in \mathbf{EL}$. For any coalition C of agents, C has a winning strategy profile in G for Win_ϕ^G if and only if it has a winning strategy profile in G' for $Win_\phi^{G'}$.*

Proof. Let f be an isomorphism between the unfoldings of G and G'. It induces a bijection between plays and a bijection between strategies. For every play π, we have that $\pi \in Win_\phi^G$ if and only if $f(\pi) \in Win_\phi^{G'}$, because isomorphic histories satisfy the same epistemic formulas. As a result the bijection between strategies also preserves winning strategies between G and G'. □

Let us now define our controller synthesis problem.

Definition 3.4.3 (The controller synthesis problem)**.**

- *Input: A DEL C-E-arena $\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle$ and an objective $\phi \in \mathbf{EL}$*
- *Output: yes if there exists a winning strategy for Controller in $G^{\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle}$ for the winning condition Win_ϕ; no otherwise.*

Remark 2. *Formally, let us define and study the problem of existence of a strategy. One can take the liberty to call the problem "controller synthesis" because all the algorithms provided can produce a winning*

strategy whenever there exists one. The same remark applies to the distributed strategy synthesis problem defined in the next section.

As the plan existence problem reduces to the controller synthesis problem (by taking an environment whose only action is the trivial one), the undecidability of the former entails that of the latter.

Theorem 3.4.1. *The controller synthesis problem is undecidable.*

Next, it will be established that in all known cases where the plan existence problem is decidable, so is the controller synthesis problem.

3.4.2 The case of non-expanding action models

Let us consider *non-expanding* action models, where actions do not expand epistemic models when executed. Typical examples of such actions are public announcements and public actions. When actions are non-expanding the search space is finite, and thus the problem is decidable. Let us establish the precise computational complexity of the problem in the case of public announcements and public actions.

Beforehand, let us provide a semantic definition of non-expanding action models, and show that they correspond to the syntactic notion of separable action models from [44]. In the following, we say that a world u in an epistemic model $\mathcal{M} = (W, (\preccurlyeq_a)_{a \in Agt}, \lambda)$ is *connected* to a world w if u can be reached from w by a sequence of epistemic relations, i.e., $(w, u) \in (\cup_{a \in Agt} \preccurlyeq_a)^*$, and the *connected component*

of w is the set of worlds connected to w. Connected actions and connected components in action models are defined similarly.

Definition 3.4.4. *An action model $\mathcal{A}$ is* non expanding *if, for every pointed epistemic model $(\mathcal{M}, w)$ and action $\alpha \in \mathcal{A}$ such that $\mathcal{M}, w \models pre(\alpha)$, the connected component of (w, α) is no bigger than the connected component of w.*

In other words, an action model is non expanding if applying any of its actions in any world of any epistemic model does not increase the number of possible worlds, modulo disconnected worlds that are irrelevant for the evaluation of epistemic properties. For instance, since their connected components are singletons, public actions and public announcements are non expanding.

Let us now refine the notion of separable action model from [44].

Definition 3.4.5. *An action model $\mathcal{A}$ is* separable *if for any two distinct events $\alpha, \beta \in \mathcal{A}$ in the same connected component, $pre(\alpha) \wedge pre(\beta)$ is unsatisfiable.*

With respect to the definition from [44], the requirement is limited to only connected pairs of events. Intuitively, disconnected events give rise to disconnected worlds when the action model is applied to an epistemic model, and thus cannot cause the connected component of interest to grow.

Proposition 3.4.1. *An action model is non expanding if and only if it is separable.*

Proof. First, let $\mathcal{A} = (A, (\preccurlyeq_a^{\mathcal{A}})_{a \in Agt}, pre, post)$ be a separable action model. Let $\alpha \in A$, and let $(\mathcal{M}, w)$ be a pointed epistemic model such that $\mathcal{M}, w \models pre(\alpha)$. For every world $(u, \beta) \in \mathcal{M} \otimes \mathcal{A}$ connected to (w, α), by definition of the update product, we have that β is connected to α. Because $\mathcal{A}$ is separable, two actions β, β' connected to α cannot be executable in a same world u. As a result, each world u connected to w can give rise to at most one world (u, β) connected to (w, α), hence $\mathcal{A}$ is non-expanding.

Conversely, if an action model $\mathcal{A} = (A, (\preccurlyeq_a^{\mathcal{A}})_{a \in Agt}, pre, post)$ is not separable, then there are two actions β, β' in a same connected component (say, the connected component of action α) such that $pre(\beta) \wedge pre(\beta')$ is satisfiable. Since β and β' are connected to α, there exist $k \geq 0$ actions $\beta_1, \ldots, \beta_k$ and $\beta'_1, \ldots, \beta'_{k'}$ and agents $a_1, \ldots, a_{k+1}$ and $a'_1, \ldots, a'_{k'+1}$ s. t. $\alpha \preccurlyeq_{a_1}^{\mathcal{A}} \beta_1 \ldots \beta_k \preccurlyeq_{a_{k+1}}^{\mathcal{A}} \beta$ and $\alpha \preccurlyeq_{a'_1}^{\mathcal{A}} \beta'_1 \ldots \beta'_k \preccurlyeq_{a'_{k'+1}}^{\mathcal{A}} \beta'$. Let us define an epistemic model $\mathcal{M} = (W, (\preccurlyeq_a)_{a \in Agt}, \lambda)$ with two chains of worlds reflecting the structure of the above two chains of actions, except that they end in the same world u that satisfies both $pre(\beta)$ and $pre(\beta')$ ($W = \{w, u_1, \ldots, u_k, u'_1, \ldots, u'_{k'}, u\}$). Accessibility relations are s. t. $w \preccurlyeq_{a_1} u_1 \ldots u_k \preccurlyeq_{a_{k+1}} u$ and $w \preccurlyeq_{a'_1} u'_1 \ldots u'_{k'} \preccurlyeq_{a'_{k+1}} u$, and the valuation function is such that w satisfies $pre(\alpha)$, each u_i satisfies $pre(\beta_i)$, each u'_i satisfies $pre(\beta'_i)$ (this is possible since actions have satisfiable preconditions), and u satisfies $pre(\beta) \wedge pre(\beta')$. The connected component of (w, α) in $\mathcal{M} \otimes \mathcal{A}$ con-

tains $\{(w, \alpha), (u_1, \beta_1), \ldots, (u_k, \beta_k), (u'_1, \beta'_1), \ldots, (u'_{k'}, \beta'_{k'}), (u, \beta), (u, \beta')\}$, which is bigger than W, the connected component of w. □

In the following, the complexity of the controller synthesis problem for non-expanding actions will be established. First, the particular case of public announcements will be considered, for which the complexity is shown to be lower than for arbitrary non-expanding actions.

Theorem 3.4.2. *When all actions are public announcements, the controller synthesis problem is* PSPACE*-complete.*

Proof. For the upper bound, it is used the fact that applying public announcements to epistemic models only removes worlds, and does not change those that remain. As a result, the number of successive public announcements to consider can be bounded by the number of worlds in the initial epistemic model. The problem can be thus solved with an alternating algorithm that runs in polynomial time, guessing existentially actions of the controller and universally those of the environment. The procedure is given in Algorithm 4.

In conclusion, let us recall that alternating polynomial time corresponds to deterministic polynomial space [43]. Note that checking epistemic formulas (preconditions and ϕ) in epistemic models, and thus also computing the update product, can be performed in polynomial time. PSPACE-hardness is proved with a polynomial reduction from TQBF (True Quantified Boolean Formulae), a PSPACE-complete

Algorithm 4 Alternating algorithm for deciding in polynomial-time the controller synthesis problem when actions are public announcements.

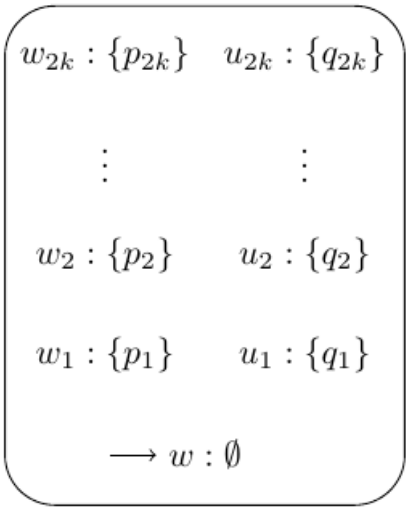
universally choose $w_\iota \in W_\iota$,
set $\mathcal{M}_{cur}, w_{cur} := \mathcal{M}, w_\iota$ as the current pointed epistemic model;
for $i := 0$ to the number of worlds in $\mathcal{M}$ **do**
  if $\mathcal{M}_{cur}, w_{cur} \models \phi$ **then**
    accept
  end if
  if i is even **then**
    existentially choose $\alpha \in A_{\text{ctr}}$ such that $\mathcal{M}_{cur}, w_{cur} \models pre(\alpha)$ (fail if no such action exists);
  end if
  if i is odd **then**
    universally choose $\alpha \in A_{\text{env}}$ such that $\mathcal{M}_{cur}, w_{cur} \models pre(\alpha)$ (fail if no such action exists);
  end if
  set $\mathcal{M}_{cur}, w_{cur} := \mathcal{M}_{cur} \otimes \mathcal{A}, (w_{cur}, \alpha)$
end for
reject

problem [119]. Given a QBF formula

$$\Phi = \exists p_1 \forall p_2 \ldots \exists p_{2k-1} \forall p_{2k} \chi(p_1, \ldots, p_{2k})$$

one can construct the following instance of the controller existence problem:

- $\mathcal{M}$ is the pointed Kripke model made up of a $\{p_i\}$-world (i.e., a world where only p_i holds) and a $\{q_i\}$-world (where q_i is an atomic proposition different from p_j for all j, and different from q_j for $j \neq i$) for each $i \in \{1, \ldots, 2k\}$, and an extra $\emptyset$-world w which is the initial pointed world; the epistemic relation for agent a is universal;



- The possible announcements are

$$\varphi_{\neg p_i} = \bigwedge_{j=1}^{i-1} K_a \neg q_j \wedge \bigwedge_{j=i}^{2k} \hat{K}_a q_j \wedge \neg p_i \wedge \neg q_i \quad \text{and}$$
$$\varphi_{p_i} = \bigwedge_{j=1}^{i-1} K_a \neg q_j \wedge \bigwedge_{j=i}^{2k} \hat{K}_a q_j \wedge \neg q_i$$

  for $i \in \{1, \ldots, 2k\}$. They belong to the controller when i is odd, and to the environment when i is even;

- The goal is $\bigwedge_{j=1}^{2k} K_a \neg q_j \wedge \chi(\hat{K}_a p_1, \ldots, \hat{K}_a p_{2k})$.

In the model $\mathcal{M}$, worlds w_i are used to encode assignments of truth values to atoms p_i: removing world w_i means setting p_i to true, while keeping it means setting p_i to false. Worlds u_i, bearing atoms q_i, are used to enforce that the value of each atom p_i is set exactly once. In announcements ϕ_{p_i} and $\phi_{\neg p_i}$, conjunct $\bigwedge_{j=1}^{i-1} K_a \neg q_j \wedge \bigwedge_{j=i}^{2k} \hat{K}_a q_j$ implies that worlds $u_1, \ldots, u_{i-1}$ have already been removed, while worlds $u_i, \ldots, u_{2k}$ are still in the model. Thus announcements ϕ_{p_i} and $\phi_{\neg p_i}$ are possible in round i, and only there.

Now observe that announcement $\phi_{\neg p_i}$, because of conjunct $\neg p_i \wedge \neg q_i$, removes both world w_i and world u_i, thus setting p_i to true. Announcement ϕ_{p_i} instead removes only world u_i, thus setting p_i to true.

In the goal formula, $\bigwedge_{j=1}^{2k} K_a \neg q_j$ means that all the variables $p_1, \ldots, p_{2k}$ have been assigned. The clause $\chi(\hat{K}_a p_1, \ldots, \hat{K}_a p_{2k})$ is the formula

$\chi(p_1, \ldots, p_{2k})$ in which we replaced p_i by $\hat{K}_a p_i$, which holds if and only if world w_i has not been removed by announcements, i.e., if and only if announcement ϕ_{p_i} was chosen at round i.

The fact that the announcements that assign values to $p_1, p_3, \ldots$ are assigned to the controller and that the announcements that assign values to $p_2, p_4, \ldots$ are played by the environment reflects the alternation of quantifiers in the formula $\exists p_1 \forall p_2 \ldots \exists p_{2k-1} \forall p_{2k} \chi(p_1, \ldots, p_{2k})$.

□

Let us now move to the class of non-expanding actions.

Theorem 3.4.3. *The controller synthesis problem for non-expanding actions is* EXPTIME*-complete.*

Proof. As for public announcements, applying a non-expanding action in a model does not add worlds. However it may change facts in worlds, so that sequences of actions of linear length may not suffice. Nonetheless, linear space is enough to store the current pointed epistemic model, and we can turn the alternating algorithm from the proof of Theorem 3.4.2 into one that runs in polynomial space. The new algorithm is given in Figure 5, in which we do not bound the length of the sequence of actions. Note that the algorithm may not terminate, but it is folklore that we can add a counter to ensure termination while staying in polynomial-space.

The EXPTIME-membership of the problem follows from the fact that alternating polynomial space corresponds to exponential time [43].

Algorithm 5 Alternating algorithm for deciding in polynomial-space the controller synthesis problem when actions are non-expanding.

universally choose $w_\iota \in W_\iota$;

set $\mathcal{M}_{cur}, w_{cur} := \mathcal{M}, w_\iota$ as the current pointed epistemic model;
$i := 0$;
while $\mathcal{M}_{cur}, w_{cur} \not\models \phi$ **do**
  if i is even **then**
    existentially choose $\alpha \in A_{\text{ctr}}$ such that $\mathcal{M}_{cur}, w_{cur} \models pre(\alpha)$(fail if no such action exists);
  end if
  if i is odd **then**
    universally choose $\alpha \in A_{\text{env}}$ such that $\mathcal{M}_{cur}, w_{cur} \models pre(\alpha)$ (fail if no such action exists);
    set $\mathcal{M}_{cur}, w_{cur} := \mathcal{M}_{cur} \otimes \mathcal{A}, (w_{cur}, \alpha)$;
    $i := 1 - i$;
  end if
end while
Accept

EXPTIME-hardness is obtained by reduction from the conditional planning problem, a variant of classical planning with full observability and non-deterministic actions, where the plan should lead to a situation satisfying the goal no matter how nondeterminism is resolved. However the plan can depend on how nondeterminism is resolved, hence the name "conditional plan" [78, 113].

Stated in our terms, conditional planning essentially corresponds to a particular case of controller synthesis which is purely Boolean (no epistemic content), but where actions chosen by the controller have nondeterministic effects, and the environment resolves nondeterminism. Since everything is purely Boolean, the initial situation is a one-

world epistemic model, i.e., a valuation over a finite set of atoms AP, the goal is a Boolean formula over AP, and each action is a one-state action model with nondeterministic postcondition. A conditional plan is then a winning strategy for the controller. Thus, to finish the reduction, one only have to show how to simulate nondeterministic actions in our setting.

In [78, 113], a nondeterministic action is modelled as a tuple $\langle \phi, \overrightarrow{post} \rangle$, where ϕ is a Boolean precondition, and $\overrightarrow{post}$ is a finite set of postconditions $\{post_1, \ldots, post_n\}$. The idea is that in each round the controller chooses an action among those whose precondition is true, and the environment resolves the non-determinism by choosing which postcondition of $\overrightarrow{post}$ to apply to the current valuation. For each nondeterministic action $\langle \phi, \{post_1, \ldots, post_n\} \rangle$ of the conditional planning instance, one can create one action for the controller that stores in a finite-domain variable *action* which action has been played, and n actions for the environment that correspond to the different possible postconditions. The action for the controller is defined as follows:

$$pre : \phi$$
$$post : action := \langle \phi, \overrightarrow{post} \rangle$$

while the actions for the environment are, for each $i \in \{1, \ldots, n\}$,

$$pre : action = \langle \phi, \overrightarrow{post} \rangle$$
$$post : post_i$$

Accessibility relations are all the identity relation, so that all actions are public actions, and the action model is non-expanding. The goal is the same as in the conditional planning instance. This finishes the proof, and also shows how the controller synthesis problem subsumes conditional planning. We now present an alternative proof that reduces from a more basic decision problem called G_4, introduced by Chandra and Stockmeyer [123]. This is essentially an adaptation of the proof from [78] for the EXPTIME-hardness of conditional planning.

The input to the G_4 problem is a 13-DNF formula over $2k$ atomic propositions $p_1, \ldots, p_k,\ q_1, \ldots, q_k$ and an initial valuation. Atoms $p_1, \ldots, p_k$ are controlled by the controller (the existential player) while $q_1, \ldots, q_k$ are controlled by the environment (the universal player). Now, the following game is played: each player, when it is his turn to play, flips the assignment of one of the variables he controls, and turns alternate. The game stops when the 13-DNF formula becomes true, and the winner is the player that made the last move. An instance of the G_4 problem is positive if the controller has a winning strategy.

We construct the following instance of our controller synthesis problem. The initial epistemic model is made up of one world, whose valuation is the initial valuation of G_4. Actions of the controller are:

$$pre : \top$$

$$post : p_1 := \neg p_1$$

$$pre : \top$$

$$post : p_k := \neg p_k$$

Actions of the environment are:

$$pre : \top$$

$$post : q_1 := \neg q_1$$

$$pre : \top$$

$$post : q_k := \neg q_k$$

Again, all accessibility relations are the identity relation. The goal is the 13-DNF formula. □

3.4.3 The case of propositional action models

To solve our controller synthesis problem in the case of actions where all preconditions and postconditions are propositional formulas, we rely on the approach followed in [87] to solve the plan existence problem for such actions. This approach has two main ingredients: (I1) when $\mathcal{A}$ is propositional, the generated structure $\mathcal{MA}^*$ can be represented finitely, and (I2) one can decide the existence of a winning strategy in a certain class of two-player games with epistemic objectives.

Theorem 3.4.4. *When action models are propositional, the controller synthesis problem is decidable, and in* $(k+1)$-EXPTIME *if the objective's modal depth is bounded by* k.

The rest of this section is devoted to prove Theorem 3.4.4. The proof amounts to showing that one can reduce the problem to the existence of a winning strategy in a certain class of games that is known to be decidable. More precisely, [87] considers turn-based games with two perfect-information players and n observers, and show that the existence of a winning strategy for one of the players is decidable for objectives expressed in a rich epistemic temporal logical language. In particular it can express the reachability of an epistemic objective $\phi \in \mathbf{EL}$. Let us call *C-E-epistemic arena* a turn-based arena with two players called *Controller* and *Environment*, and where agents in *Agt* are observers.

Theorem 3.4.5 ([39])**.** *The existence of a winning strategy for Controller in a finite C-E-epistemic arena G for an epistemic reachability winning condition Win_ϕ where $\phi \in \mathbf{EL}$ is of modal depth k can be decided in time k-exponential in $|G|$ and $|\phi|$.*

Remark 3. *To be precise the variant of the problem studied in [39] does not consider multiple possible initial states as it has been done here, but only one. However, the result can be extended by a straightforward reduction from the case of multiple initial states to the case of a single one. This can be done by adding an artificial unique initial position controlled by Environment that branches to all real initial positions, and requires to reach the goal after at least one step (this is expressible in the logic considered in [39]).*

Note that the arena $G^{\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle}$ used in Definition 3.4.2 to define the controller synthesis problem is a C-E-epistemic arena with infinitely many positions. One can show that if $\mathcal{A}$ is propositional, then one can construct a *finite* C-E-epistemic arena G of exponential size that is equivalent to $G^{\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle}$. Since arena equivalence preserves winning strategies for winning condition Win_ϕ (Lemma 3.4.1), and we can solve the problem in k-exponential time on finite C-E-epistemic arenas (Theorem 3.4.5), the $k+1$-EXPTIME upper bound of Theorem 3.4.4 follows.

Proposition 3.4.2. *Given a DEL C-E-arena $\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle$ where $\mathcal{A}$ is propositional, one can construct a finite C-E-epistemic game arena G equivalent to $G^{\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle}$ and of size $|G| \leq |\mathcal{M}| + |\mathcal{A}| \times 2^{m+1}$, where m is the number of atomic propositions involved in $\mathcal{M}, \mathcal{A}$ and ϕ.*

Proof. Let $\mathcal{M} = (W, (\preccurlyeq_a)_{a \in Agt}, \lambda)$ and $\mathcal{A} = (A, (\preccurlyeq_a^{\mathcal{A}})_{a \in Agt}, pre, post)$, and let AP_i be the atomic propositions involved. The idea is that when preconditions and postconditions are all propositional, knowing the current valuation is enough to evaluate them. One can also include in positions either the world (in starting positions) or the last action seen so that one can define epistemic relations accurately. Finally, it is needed a bit to indicate who should play between *Controller* and *Environment*. As a result, a position of the game arena G that we build is either an initial world $w \in \mathcal{M}$ or a tuple $(\alpha, \nu, \mathsf{turn})$ where $\alpha \in \mathcal{A}$ represents the last action performed, $\nu \in 2^{AP_i}$ is the current valuation,

and $\mathsf{turn} \in \{0, 1\}$ indicates whose turn it is to play: 0 for *Environment* and 1 for *Controller*. In an initial position w, *Environment* can choose an action $\alpha \in A_{\text{env}}$ such that $w \models pre(\alpha)$ and move to $(\alpha, \lambda'(w, \alpha), 1)$; in a world of the form $(\alpha, \nu, \mathsf{turn})$, if $\mathsf{turn}(v) = 0$ (resp., $\mathsf{turn}(v) = 1$), *Environment* (resp, *Controller*) chooses an action $\alpha' \in A_{\text{env}}$ (resp., $\alpha' \in A_{\text{ctr}}$) such that $\nu \models pre(\alpha')$, and moves to $(\alpha', \lambda(\nu, \alpha'), 1 - \mathsf{turn})$.

Formally, letting $A = A_{\text{env}} \uplus A_{\text{ctr}}$, we define the C-E-epistemic arena $G = (Act, V, v_\iota, RAct, \delta, \preccurlyeq', \lambda')$ where

- $Act = A_{\text{env}} \uplus A_{\text{ctr}}$
- $V = W \cup (2^{AP_i} \times A \times \{0, 1\})$
- $V_\iota = W_\iota$
- Repertoires of actions are as follows:
  - $Act_{\text{Env}}(v) = \begin{cases} \{\alpha \in A_{\text{env}} \mid w \models \alpha\} & \text{if } v = w \\ \{\alpha \in A_{\text{env}} \mid \nu \models \alpha\} & \text{if } v = (\nu, \alpha, 0) \end{cases}$
  - $Act_{\text{Con}}(v) = \{\alpha \in A_{\text{ctr}} \mid \nu \models \alpha\}$ if $v = (\nu, \alpha, 1)$
  - $Act_a(v) = \{idle\}$ in all other cases
- for a joint action $\vec{\alpha}$ whose only non-idle action is α, we let:

$$\delta(v, \vec{\alpha}) = \begin{cases} (\alpha, \lambda(w, \alpha), 1) & \text{if } v = w \\ (\alpha, \lambda(\nu, \alpha), 1 - \mathsf{turn}) & \text{if } v = (\alpha', \nu, \mathsf{turn}) \end{cases}$$

- agents' epistemic relations are as follows:
  - $\preccurlyeq_{\text{Con}}$ and $\preccurlyeq_{\text{Env}}$ are the identity relation
  - for each other agent a,

$$v \preccurlyeq'_a v' \text{ if } \begin{cases} v = w,\ v' = w' \text{ and } w \preccurlyeq_a w', \text{ or} \\ v = (\nu, \alpha, \mathsf{turn}),\ v' = (\nu', \alpha', \mathsf{turn}) \text{ and } \alpha \preccurlyeq_a^{\mathcal{A}} \alpha' \end{cases}$$

- $\lambda'(v) = \begin{cases} \lambda(w) & \text{if } v = w \\ \nu & \text{if } v = (\nu, \alpha, \mathsf{turn}) \end{cases}$

The number of positions in G is $|\mathcal{M}| + |\mathcal{A}| \times 2^{m+1}$. One can then check that the unfoldings of G and that of $G^{\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle}$ are isomorphic.

□

3.5 Distributed Strategy Synthesis

In classic DEL epistemic planning, one external entity (the Planner) chooses a plan to achieve a goal, while all the epistemic agents in the DEL models are passive observers. In the controller synthesis problem introduced and studied in the previous section, the Planner is called the *Controller* and is now playing against the *Environment*, but the agents are still mere observers. Let us now go one step further in generalizing the problem and make the agents themselves choose the actions that are performed: they become players. Naturally, and consistently

with the fundamentals of games with imperfect information [100], it is ensured that their strategies are consistent with their knowledge.

In this setting, it will be consideres the problem of *distributed strategy synthesis*, which consists in deciding whether a designated subset of agents (a *coalition*) has a distributed strategy (or strategy profile) to enforce some objective against the other agents. To this end, the set Agt of agents is split into two teams $Agt_\exists$ and $Agt_\forall$ that play against each other.

3.5.1 Setting up the game

As discussed in Section 3.3.2, when considering players with imperfect information we assume that their epistemic relations are equivalence relations. Since all agents are now players, it will be assumed that all epistemic and action models are S5, i.e., all their epistemic relations are equivalence relations.

Let us fix an initial epistemic model $\mathcal{M} = (W, (\approx_a)_{a\in Agt}, \lambda)$ and an action model $\mathcal{A} = (A, (\approx_a^{\mathcal{A}})_{a\in Agt}, pre, post)$ whose set of actions A is partitioned into subsets $(A_a)_{a\in Agt}$ of actions for each player. Let us describe a multiplayer game arena of imperfect information based on these models. For the moment, let us define turn-based games because the DEL update product captures the occurrence of a single event at a time. In Section 3.6, it will be introduced a novel update product that will be used to define concurrent DEL games.

To model turns, let us use a variable turn ranging over Agt to represent whose turn it is to play. We require that an action controlled by agent a is executable only when it is agent a's turn to play: for each $\alpha \in A_a$, $pre(\alpha)$ implies $\mathsf{turn} = a$. We also require that postconditions for variable turn do not depend on the current world, but instead the next value of turn is completely determined by the action only: for every action α and agent a, $post(\alpha)(\mathsf{turn} = a) \in \{\top, \bot\}$ (recall that such variables with finite domain can be encoded with atomic propositions, see sections 3.3.3 and 3.3.3).

Moreover, in order to obtain a proper imperfect-information game, the following hypotheses are required:

Hypotheses on $\mathcal{M}$ and $\mathcal{A}$

(H1) **The starting player is known:** there is a player a such that for all $w \in W$, $\mathcal{M}, w \models \mathsf{turn} = a$;

(H2) **The turn stays known:** for all actions α, α' and agent a, if $\alpha \approx_a^{\mathcal{A}} \alpha'$, then α and α' assign the same value to turn.

(H3) **Players know their available actions:** if $w\alpha_1 \dots \alpha_n \models \mathsf{turn} = a$ and $w\alpha_1 \dots \alpha_n \approx_a w'\alpha'_1 \dots \alpha'_n$, then the same actions of a are executable in both worlds: for all $\alpha \in A_a$, $w\alpha_1 \dots \alpha_n \models pre(\alpha)$ iff $w'\alpha'_1 \dots \alpha'_n \models pre(\alpha)$.

Remark 4. *Hypothesis (H1) and (H2) concern syntactic aspects of arena and are therefore easy to check. Regarding Hypothesis (H3), our*

results split in two cases: In Section 3.5.4, we allow action models to have modal preconditions, so that imposing the precondition of each α, say in A_a, to be of the form $K_a\phi$ makes Hypothesis (H3) true. On the contrary in Section 3.5.5, the action model $\mathcal{A}$ is propositional. By Proposition 3.4.2, the generated structure $\mathcal{MA}^*$ can be represented by some finite C-E-arena G. By letting Controller play alone, i.e., by letting $V_{Con} = V$ and $V_{Env} = \emptyset$, we obtain a game where every strategy of Controller induces a unique outcome, and every possible play in G is the outcome of one such strategy. Consider formula:

$$\phi_{H3} := \bigwedge_{a \in Agt} \Big[\mathsf{turn} = a \to \big(\bigwedge_{\alpha \in A_a} pre(\alpha) \to K_a pre(\alpha) \big) \Big]$$

It expresses that the agent in control of the current position knows which of its actions are executable. As a result, Controller has a winning strategy for objective $Win_{\neg\phi_{(H3)}}$ if and only if (H3) does not hold, and by Theorem 3.4.5 one can decide this.

3.5.2 The distributed strategy synthesis problem

Let us now define formally the problem of distributed strategy synthesis for DEL games. The first step is to define DEL turn-based game arenas with imperfect information, that will be simply called DEL arenas in this section.

Definition 3.5.1. *A* DEL arena $\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle$ *consists of an initial*

multipointed epistemic model $(\mathcal{M}, W_\iota)$ *and an action model* $\mathcal{A}$ *that satisfy (H1), (H2) and (H3).*

Given a DEL game arena $\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle$, one can define an infinite turn-based game arena $G^{\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle}$ where all agents are players with imperfect information.

Definition 3.5.2. *Given a DEL arena* $\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle$*, it is possible to define the deterministic turn-based arena* $G^{\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle}$ *as the tuple* $(Act, V, V_\iota, RAct, \delta, \approx, \lambda)$ *where, writing* $A = \uplus_{a \in Agt} A_a$ *for the sets of actions in* $\mathcal{A}$ *and* $\mathcal{MA}^n = (W^n, (\approx_a^n)_{a \in Agt}, \lambda^n)$ *for every* n*:*

- *the set of actions is* $Act = A \uplus \{idle\}$*;*
- *the set of positions is* $V = \cup_{n \in \mathbb{N}} W^n$*, partitioned as follows: for each* $a \in Agt$*,* $V_a = \{v \mid \mathcal{MA}^*, v \models (\mathsf{turn} = a)\}$*;*
- *the initial positions are* $V_\iota = W_\iota$*,*
- *for each agent* $a \in Agt$*, its repertoire of actions is defined as follows:*

$$Act_a(v) = \begin{cases} \{\alpha \in A_a \mid v \models pre(\alpha)\} & \text{if } v \in V_a \\ \{idle\} & \text{otherwise;} \end{cases}$$

- $\delta(v, \vec{\alpha}) = v \cdot \alpha$*, where* α *is the only non-idle action in* $\vec{\alpha}$*;*
- *for each agent* $a \in Agt$*,* $\approx_a = \cup_n \approx_a^n$*;*

- $\lambda(v) = \lambda^n(v)$, *where* n *is such that* $v \in W^n$.

This is a well-defined turn-based game: thanks to H1 and H2, players always know whose turn it is, and thanks to H3, players always know their available actions.

Definition 3.5.3 (Distributed strategy synthesis problem)**.**

- *Input: DEL arena* $\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle$*, a team* $Agt_\exists \subseteq Agt$ *and a goal* $\phi \in \mathbf{EL}$*;*
- *Output: yes if team* $Agt_\exists$ *has a winning strategy in* $G^{\langle \mathcal{M}, W_\iota, \mathcal{A} \rangle}$ *for winning condition* Win_ϕ*, no otherwise.*

One may write $Agt_\forall = Agt \setminus Agt_\exists$ for the opposite team.

Let us first establish an undecidability result for this problem. Unlike the controller synthesis problem which we proved decidable for propositional actions, synthesising distributed strategies is undecidable for propositional actions, already for a team of two players.

3.5.3 Undecidability for two existential players

The following Theorem 3.5.1 is a reformulation in our setting of the classical undecidability result from Reif and Peterson [108]. However, it is promoted an existing elegant reformulation of that very same result, called TEAM DFA GAME [45, Def. 1,p. 14:7], that can be reduced to our distributed strategy synthesis problem.

Theorem 3.5.1. *The distributed strategy synthesis problem is undecidable, already for a propositional action model and two existential players against one universal player.*

Proof. The proof is given by reduction from the problem TEAM DFA GAME [45, Def. 1, p. 14:7], shown to be undecidable. Let us consider a two-versus-one (players a and b versus player $\forall$) team game played on a deterministic finite automaton (DFA) $\mathtt{A}$ whose alphabet is $\{0, 1\}$, whose set of states is Q, initial state is q_0, transition function δ. Special subsets of states $F_\exists$ and $F_\forall$ are given. The game starts with $\mathtt{A}$ being in state q_0. Each round is divided in six steps:

1. if the current state q is in $F_\exists$ then team $\{a, b\}$ wins; if the current state q is in $F_\forall$ then team $\{\forall\}$ wins;
2. Player $\forall$ inputs two bits β, β' into $\mathtt{A}$;
3. Player a learns β;
4. Player a inputs one bit m into $\mathtt{A}$;
5. Player b learns β';
6. Player b inputs one bit m' into $\mathtt{A}$.

At each step, player $\forall$ has perfect information. TEAM DFA GAME is the decision problem: given an DFA $\mathtt{A}$, subsets of states $F_\exists$, $F_\forall$, does the team $\{a, b\}$ have a winning strategy?

The rest of the proof consists in representing the initial situation, the game rules and the goal of a TEAM DFA GAME instance as a distributed strategy synthesis problem instance.

Definition of the reduction. Let $(\mathtt{A}, F_\exists, F_\forall)$ be an instance of TEAM DFA GAME. Teams are $Agt_\exists = \{a, b\}$ and $Agt_\forall = \{\forall\}$.

We introduce a finite-domain variable q that ranges over the set of states of $\mathtt{A}$. The variable q can be represented by a finite set of atomic propositions: for example, for an automaton with 8 states from $\{0, \ldots, 7\}$, three atomic propositions, $bit_1(q)$, $bit_2(q)$ and $bit_3(q)$ so that say $(q = 5)$ is the Boolean formula $bit_1(q) \wedge \neg bit_2(q) \wedge bit_3(q)$.

Let us also introduce a finite-domain variable stp that ranges over $\{1, 2, 3, 4, 5, 6\}$. The Boolean variable lost is true if the team $Agt_\exists$ has lost. Let us define $\mathcal{M}, w$ to be the single-world S5 epistemic model in which $\mathsf{turn} = \forall$, $q = q_0$, $\mathsf{stp} = 1$, $\neg\mathsf{lost}$. The actions in $A_\forall$ form an a- and b-indistinguishably equivalence class and are of the form:

- $pre : \mathsf{turn} = \forall \wedge \mathsf{stp} = 1 \wedge q{\in}F_\forall$; $post : \mathsf{lost}{:=}\top, \mathsf{stp}{:=}2$
- $pre : \mathsf{turn} = \forall \wedge \mathsf{stp} = 1 \wedge q{\notin}F_\forall$; $post : \mathsf{stp}{:=}2$
- $pre : \mathsf{turn} = \forall \wedge \mathsf{stp} = 2$;
  $post : \beta{:=}0, \beta'{:=}0, q{:=}\delta(q, 00), \mathsf{turn}{:=}a, \mathsf{stp}{:=}3$
- $pre : \mathsf{turn} = \forall \wedge \mathsf{stp} = 2$;
  $post : \beta{:=}1, \beta'{:=}0, q{:=}\delta(q, 10), \mathsf{turn}{:=}a, \mathsf{stp}{:=}3$

- $pre : \mathsf{turn} = \forall \wedge \mathsf{stp} = 2;$

  $post : \beta{:=}0, \beta'{:=}1, q{:=}\delta(q, 01), \mathsf{turn}{:=}a, \mathsf{stp}{:=}3$

- $pre : \mathsf{turn} = \forall \wedge \mathsf{stp} = 2;$

  $post : \beta{:=}1, \beta'{:=}1, q{:=}\delta(q, 11), \mathsf{turn}{:=}a, \mathsf{stp}{:=}3$

A_a is a b-indistinguishably equivalence class and contains:

- $pre : \mathsf{turn} = a \wedge \mathsf{stp} = 3 \wedge \beta; post : \mathsf{stp} = 4$

- $pre : \mathsf{turn} = a \wedge \mathsf{stp} = 3 \wedge \neg\beta; post : \mathsf{stp} = 4$

- $pre : \mathsf{turn} = a \wedge \mathsf{stp} = 4;$

  $post : m{:=}\bot, \mathsf{stp}{:=}5, q{:=}\delta(q, 0), \mathsf{turn}{:=}b$

- $pre : \mathsf{turn} = a \wedge \mathsf{stp} = 4;$

  $post : m{:=}\top, \mathsf{stp}{:=}5, q{:=}\delta(q, 1), \mathsf{turn}{:=}b$

A_b is an a-indistinguishably equivalence class and contains:

- $pre : \mathsf{turn} = b \wedge \mathsf{stp} = 5 \wedge \beta'; post : \mathsf{stp}{:=}6$

- $pre : \mathsf{turn} = b \wedge \mathsf{stp} = 5 \wedge \neg\beta'; post : \mathsf{stp}{:=}6$

- $pre : \mathsf{turn} = b \wedge \mathsf{stp} = 6;$

  $post : m'{:=}\bot, \mathsf{stp}{:=}1, q{:=}\delta(q, 0), \mathsf{turn}{:=}\forall$

- $pre : \mathsf{turn} = b \wedge \mathsf{stp} = 4;$

  $post : m'{:=}\top, \mathsf{stp}{:=}1, q{:=}\delta(q, 1), \mathsf{turn}{:=}\forall$

The assignments $q := \delta(q, 0)$ and $q := \delta(q, 1)$ are shortcuts for some "if statements" on states, e.g. 'if $q = 5$, then $q := 2$'. For instance, assuming that we have eight states $\{s_0, \ldots, s_7\}$ which are thus representable with three bits, the assignment $q := \delta(q, 0)$ is simulated by the following set of propositional assignments: $\{bit_i(q) := \psi_i\}_{i=1..3}$, where ψ_i is the Boolean formula $\bigvee_{k \in 0..7 \text{ s.t. the i-th bit of } \delta(s_k,0) \text{ is } 1}(q = s_k)$.

The goal formula ϕ is $\neg \mathsf{lost} \wedge \mathsf{stp} = 1 \wedge (q \in F_\exists)$. □

Let us now turn to decidable cases: games with imperfect information and epistemic objectives are known to be decidable either when actions are public [26], or when information is hierarchical [88]. Similar results in this setting are established.

3.5.4 The case of non-expanding action models

Theorems 3.4.2 and 3.4.3 of Section 3.4 generalise to the problem of distributed strategy synthesis. First, we inherit the lower bounds by noticing that in the proofs of both theorems, the reductions used to establish the lower-bounds yield DEL games where *Controller* and *Environment* alternate turns; by letting $Agt_\exists = \{controller\}$, we obtain instances of the distributed strategy synthesis problem. Second, the upper bounds are obtained by adapting the alternating algorithms for the upper bounds of Theorems 3.4.2 and 3.4.3. We need to ensure that existential choices of actions of an agent $a \in Agt_\exists$ lead to a *uniform* strategy.

To do that, every time agent a picks an action α, we perform an extra universal choice over $\approx_a$-indistinguishable worlds, and continue executing the algorithm from these worlds.

Theorem 3.5.2. *For public announcements, the distributed strategy synthesis problem is* PSPACE*-complete.*

Proof. Hardness follows from Theorem 3.4.2 (consider team $Agt_\exists = \{controller\}$). To establish PSPACE membership we provide an alternating algorithm that runs in polynomial time. This algorithm is similar to the one given in the proof of Theorem 3.4.2, except that we add universal choices of $\approx_a^{\mathcal{A}}$-successors for player a in $Agt_\exists$. More precisely, the algorithm works as follows: when it is the turn of a player a in $Agt_\forall$ to play, universally guess an executable action in A_a. When it is the turn of a player a in $Agt_\exists$ to play, then perform the following steps:

- first existentially guess an executable action α in A_a;
- second, universally guess $\approx_a$-successor of the pointed world in the current epistemic model and make it as the new pointed world;
- compute the new epistemic model by executing the α.

As for Theorem 3.4.2, the length of such a sequence is bounded by the number of worlds in the initial epistemic model $\mathcal{M}$. The obtained algorithm is given in Algorithm 6.

Algorithm 6 Alternating algorithm for deciding in polynomial-time the distributed strategy synthesis problem when actions are public announcements.

set $\mathcal{M}_{cur}, w_{cur} := \mathcal{M}, w_\iota$ as the current pointed epistemic model;
for $i := 0$ to the number of worlds in $\mathcal{M}$ **do**
  if $\mathcal{M}_{cur}, w_{cur} \models \phi$ **then**
    accept
  end if
  let a be the agent such that $\mathcal{M}_{cur}, w_{cur} \models \mathsf{turn} = a$;
  if $a \in Agt_\exists$ **then**
    existentially choose $\alpha \in A_a$ such that $\mathcal{M}_{cur}, w_{cur} \models pre(\alpha)$
    (fail if no such action exists);
    universally choose u such that $w_{cur} \approx_a u$;
    set $w_{cur} := u$;
  else
    universally choose $\alpha \in A_a$ such that $\mathcal{M}_{cur}, w_{cur} \models pre(\alpha)$
    (fail if no such action exists);
  end if
  set $\mathcal{M}_{cur}, w_{cur} := \mathcal{M}_{cur} \otimes \mathcal{A}, (w_{cur}, \alpha)$;
end for
reject

□

Theorem 3.5.3. *For public actions, the distributed strategy synthesis problem is* EXPTIME*-complete.*

Proof. The alternating algorithm that runs in polynomial space is similar to the one given in the proof of Theorem 3.4.2. The algorithm is in polynomial space for the same reason said in the proof of Theorem 3.4.3. The obtained procedure is given in Algorithm 7.

Algorithm 7 Alternating algorithm for deciding in polynomial-space the distributed strategy synthesis problem when actions are non expanding.

set $\mathcal{M}_{cur}, w_{cur} := \mathcal{M}, w_\iota$ as the current pointed epistemic model;
while $\mathcal{M}_{cur}, w_{cur} \not\models \phi$ **do**
  let a be the agent such that $\mathcal{M}_{cur}, w_{cur} \models \mathsf{turn} = a$;
  if $a \in Agt_\exists$ **then**
    existentially choose $\alpha \in A_a$ such that $\mathcal{M}_{cur}, w_{cur} \models pre(\alpha)$ (fail if no such action exists);
    universally choose a world u such that $w_{cur} \approx_a u$;
    set $w_{cur} := u$;
  else
    universally choose $\alpha \in A_a$ such that $\mathcal{M}_{cur}, w_{cur} \models pre(\alpha)$ (fail if no such action exists);
  end if
  set $\mathcal{M}_{cur}, w_{cur} := \mathcal{M}_{cur} \otimes \mathcal{A}, (w_{cur}, \alpha)$;
end while
accept

Hardness follows from Theorem 3.4.3. □

Let us now turn to a decidable case for propositional actions.

3.5.5 Propositional actions+hierarchical information

Let us now consider the case of propositional action models. Unlike public announcements or public actions, they may make the size of epistemic models grow unboundedly. But by restricting to cases where the information of the different players is hierarchical, which makes it easier to synchronise the existential players' strategies, one still manage to retain decidability.

Very informally, when information is hierarchical, one can transform the game into an equivalent game of perfect information where positions are blown up to incorporate "knowledge states" of the different players in the coalition, and a meta player chooses actions for all of them. The fact that each player in the coalition knows more than the next ensures that a knowledge state of a player is never split between two knowledge states of another, which makes it possible to define the meta-positions in such a way that the final perfect-information game is equivalent to the original one (see [105] for more detail on this construction).

According to Theorem 3.5.1, the distributed strategy synthesis problem is undecidable for propositional actions and a two-player team $Agt_{\exists} = \{a, b\}$ against team $Agt_{\forall} = \{\forall\}$. Observe the proof of Theorem 3.5.1: in each round a only learns the first bit produced by $\forall$'s move, while b only learns the second bit. As a result, player a's infor-

mation is not comparable to player b's information, in the sense that there is not one who knows more than the other. This is a well-known source of undecidability in games with imperfect information (see for instance [57]). A classic restriction to regain decidability is to assume, instead, that the agents can be ordered from the one who knows the least to the one who knows the most, each one knowing at least everything that those before know. This kind of configuration is usually called *hierarchical information* [105, 109].

Given a DEL arena $\langle \mathcal{M}, w_\iota, \mathcal{A} \rangle$ and a team $Agt_\exists \subseteq Agt$, we say that there is *hierarchical information* if $Agt_\exists$ can be totally ordered $(a_1 < \ldots < a_N)$ so that $\approx_{a_i} \supseteq \approx_{a_{i+1}}$ and $\preccurlyeq^{\mathcal{A}}_{a_i} \supseteq \preccurlyeq^{\mathcal{A}}_{a_{i+1}}$, for each $1 \leq i < N$.

Theorem 3.5.4. *Distributed strategy synthesis with propositional actions and hierarchical information is decidable.*

Proof. Similarly to Proposition 3.4.2, if $\mathcal{A}$ is propositional, one can construct a finite representation of $G^{\langle \mathcal{M}, w_\iota, \mathcal{A} \rangle}$ in the form of a finite turn-based arena G, the main difference being the number of players. Since turn-based game arenas are a particular case of concurrent game arenas, and reachability of epistemic goals can be expressed in epistemic temporal logic, one can conclude by recalling that distributed strategy synthesis for epistemic temporal objectives is decidable on concurrent game structures when information among the existential players is hierarchical [88, 111] (see [91, Proposition 17, p.11] for more

detail). □

3.6 Concurrent Games

Let us now go one step further and move from turn-based games to concurrent ones. As already mentioned, the existing DEL framework cannot capture concurrent execution of actions by different agents, as its update product only models the execution of a single event at a time.

Some works consider concurrent execution of abstract actions (as in *concurrent game structures* [48]), or concurrent execution of purely epistemic actions without effects on the world ([3, 30]) or only public [75]. But to our knowledge, concurrent execution of arbitrary epistemic actions with explicit effects has never been studied. And yet they are essential for modelling realistic situations: consider the two-robot coordination example of Figure 3.2.

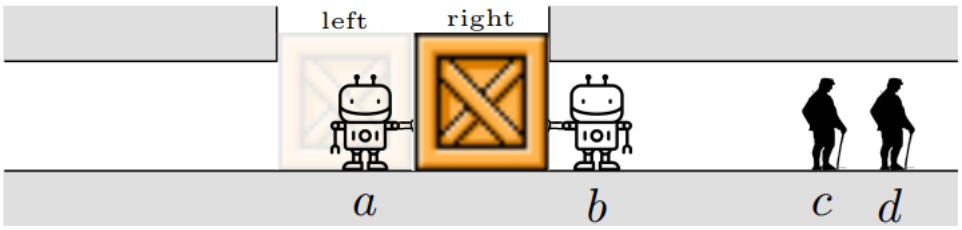


Figure 3.2: The two-robot coordination example.

Some container can be placed either left or right. Two robots (or agents), a and b, can either wait or push the container, which incidentally is energy-consuming. Both agents waiting, the container

stays in place. Otherwise, if a pushes while b waits, the container moves right (if not already there), and, symmetrically, if b pushes while a waits, it moves left (if not already there). Agents pushing concurrently results in a *conflict*.

To determine the effect of such conflicting actions, one solution is to consider that they are *blocking*, as done for instance in [58] in temporal planning, or facts involved in conflicts are just maintained [75]. Another option consists in selecting a maximal subset of non-conflicting actions, as done for instance in [55] in UML work-flow modelling. In this work, the latter solution is adopted, which is more challenging. A major additional difficulty with respect to this latter UML setting arises from epistemic features of actions of the DEL framework. In our robot example, while robots a and b perfectly perceive pushing and waiting actions, the two other agents c and d cannot distinguish them. Note that this issue is not tackled in [75] since actions are public.

The first contribution is the definition of a *DEL concurrent update product* that provides the dynamics of concurrent epistemic actions. Noticeably, this new product yields a non-deterministic dynamics controlled by a *scheduler*, whose role is to resolve conflicts. On the basis of this product, it will be shown how finite DEL presentations can generate infinite concurrent game arenas.

The second contribution is a proof that distributed synthesis can be solved for such games for two cases: the case where all actions are

public (perfectly observed by everyone), and the case where all actions have propositional pre- and postconditions and information among the agents is hierarchical (Theorems 3.6.2 and 3.6.4).

To establish this, it is needed to generalize to the concurrent setting the result stating that when pre- and post-conditions are propositional, the generated game is equivalent to a finite game arena (Proposition 3.4.2), and one can establish a similar result for public actions. Then, existing results on the model-checking problem for the epistemic strategic logic $\mathsf{ATL}^*_{\mathrm{K}}$ [127] are transferred to obtain, in particular, decidability of distributed synthesis for rich temporal epistemic objectives.

3.6.1 Concurrent Actions

Fix an epistemic state $(\mathcal{M}, w)$ and an action model $\mathcal{A}$. Next, in order to have a fine modeling of conflicts, we will distinguish between propositions that all agents can modify, and propositions that are private to an agent. From now on, atomic propositions are partitioned into *shared propositions* (AP^s) that all agents can modify, and *private* ones: AP^p_a is the set of private propositions of agent a. As a result, $AP = AP^c \uplus \biguplus_{a \in Agt} AP^p_a$, where $\uplus$ is the disjoint union. An agent can play any action that does not modify private propositions of others. We gather in set A_a those actions, namely those whose postconditions are undefined on $\cup_{b \neq a} AP^p_b$. A *joint action* is a tuple

$\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_N \rangle \in \prod_{a \in Agt} A_a$, and we let $\vec{\alpha}_b$ denote action α_b.

Finally, a joint action $\vec{\alpha}$ is *available in* w when every *individual* action $\vec{\alpha}_b$ can be executed in w: $(\mathcal{M}, w) \models pre(\vec{\alpha}_1) \wedge \cdots \wedge pre(\vec{\alpha}_n)$.

3.6.2 Conflicts

Let us define a formula $\mathsf{noconflict}(\vec{\alpha})(p)$ expressing that all individual actions of a joint action $\vec{\alpha}$ agree on their effect (if any) on proposition p. We first introduce the set $Agt(\vec{\alpha}, p)$ of agents whose individual action in $\vec{\alpha}$ has an effect on p. Formally, $Agt(\vec{\alpha}, p)$ is the set $\{a \in \{1, \ldots, N\} \mid post(\vec{\alpha}_a)(p)$ is defined$\}$.

We then define the formula

$$\mathsf{noconflict}(\vec{\alpha})(p) := \bigwedge_{a \in Agt(\vec{\alpha},p)} post(\vec{\alpha}_a)(p) \vee \bigwedge_{a \in Agt(\vec{\alpha},p)} \neg post(\vec{\alpha}_a)(p)$$

which says that all agents that act on p make it true, or they all make it false.

Now, in a situation where all individual actions of the joint action $\vec{\alpha}$ agree on their effect on each proposition, the effect of executing them concurrently can be determined unambiguously. Such a situation is expressed by the formula:

$$\mathsf{noconflict}(\vec{\alpha}) := \bigwedge_{p \in AP} \mathsf{noconflict}(\vec{\alpha})(p). \tag{3.1}$$

Let us now use this formula to define consistent, or non-conflicting,

tuples of actions.

Definition 3.6.1. *A joint action* $\vec{\alpha}$ *is* non-conflicting in w *if* $\mathcal{M}, w \models \mathsf{noconflict}(\vec{\alpha})$. *Otherwise* $\vec{\alpha}$ *is* conflicting in w.

Example 1. *In our running example,* $\langle \mathtt{push}_a, \mathtt{push}_b \rangle$ *is conflicting in both worlds* w *and* u.

Notice that an easy case that guarantees non-conflicting actions (everywhere) is the case where agents can act on disjoint subsets of atomic propositions, as it is often the case in fully distributed systems.

As said earlier, executing a non-conflicting joint action is clear, as the effect is non ambiguous. This is not the case of conflicting joint actions. One could consider that a conflicting joint action cannot be executed, leading to a blocking situation. This interpretation may be the right one in some applications.

Here, instead, a solution that tries to reduce to the minimum action blocking, and execute as much as possible of a conflicting joint action is proposed. More precisely, when the actions selected by the agents are in conflict, it is proposed to select a subset of the joint action that is conflict-free. This subset is executed, while remaining actions are *inhibited*. Here again, to maximize liveness of the system, actions that are not chosen are not completely blocked, but replaced by a *ghost* version of themselves that still can have effects, but only on the private propositions of the agent who chose the action. This ensures that no conflict is present in the actions finally executed. The only

remaining ambiguity concerns the selection of the maximal consistent subset of actions. This results in a nondeterministic product, where nondeterminism is resolved by a *scheduler*.

3.6.3 The role of the scheduler

Given a conflicting available joint action, the scheduler selects a maximal subset of consistent individual actions, and *inhibits* the remaining actions by "cancelling" their effects on shared variables. An inhibited action of agent a may still have effects on AP_a^p, which may differ from the original ones. The inhibiting mechanism is implemented by a *ghost mapping* $\circ : A \to A$ with the following properties: for every agent $a \in Agt$ and every action $\alpha \in A_a$, the *ghost* $\circ(\alpha)$ (written $\alpha_\circ$) must be in A_a and $post(\alpha_\circ)$ must be undefined on AP^s. This assumption ensures that no conflict arises when inhibiting actions. We also assume that $\alpha_\circ$ is executable as long as α is, i.e., that $pre(\alpha_\circ)$ is a logical consequence of $pre(\alpha)$.

Example 2. *For the robots,* $AP^s = \{\texttt{left}\}$, $AP_a^p = \{\texttt{lowE}_a\}$,$AP_b^p = \{\texttt{lowE}_b\}$, *and* $AP_c^p = AP_d^p = \emptyset$. *Facing the conflicting joint action* $\langle \texttt{push}_a, \texttt{push}_b \rangle$, *the scheduler can choose between* $\langle \texttt{push}_{a\circ}, \texttt{push}_b \rangle$ *and* $\langle \texttt{push}_a, \texttt{push}_{b\circ} \rangle$. *In this example, it is reasonable to set* $\texttt{push}_{a\circ} := \texttt{wait}_a$ *and* $\texttt{push}_{b\circ} := \texttt{wait}_b$.

The epistemic relations from a ghost can be set in many different ways, and the choice is a design matter. For example, assume that

ghost actions are distinct from those that can be played by agents, and that ghosts are not epistemically related to those playable actions. Then the scheduler's selection is public: indeed, when for joint action $\vec{\alpha}$ scheduler inhibits the action of agent a, all agents are implicitly informed that agent a is discarded.

Definition 3.6.2. *If* $A = \mathsf{A}_\bullet \uplus \mathsf{A}_\circ$ *is split between* playable *actions* $\mathsf{A}_\bullet$ *and ghosts ones* $\mathsf{A}_\circ$*, and* $\mathsf{A}_\bullet$ *and* $\mathsf{A}_\circ$ *are not related by any* $\approx_a$*, one can say that the scheduler is* public*.*

In the sequel, action models $\mathcal{A}$ are considered implicitly equipped with a ghost mapping.

3.6.4 Concurrent update product

Let us define the execution of a (possibly conflicting) joint action in the epistemic state $(\mathcal{M}, w)$. This is achieved by the *concurrent update product* $\boxplus$, that generalises the classic update product $\otimes$ (Definition 3.3.6) and also the proposal in [131, p.14].

Before defining this product, let us formalise the possible choices for the scheduler. We say that joint action $\vec{\beta}$ is a *sub-action* of joint action $\vec{\alpha}$, written $\vec{\beta} \preceq \vec{\alpha}$, whenever $\vec{\beta}$ is obtained from $\vec{\alpha}$ by replacing some individual actions by their ghosts. Formally, $\vec{\beta} \preceq \vec{\alpha}$ if $\vec{\beta}_a \in \{\vec{\alpha}_a, (\vec{\alpha}_a)_\circ\}$, for every $1 \leq a \leq N$. Remark that, by definition of the ghost mapping, if $\vec{\alpha}$ is available in w, so are its sub-actions.

Given a joint action $\vec{\alpha}$, we let $\texttt{Max}(\vec{\alpha}, w)$ be the set of joint actions composed of the $\preceq$-maximal sub-actions of $\vec{\alpha}$ non-conflicting in w. Formally,

$$\texttt{Max}(\vec{\alpha}, w) :=$$
$$\{\vec{\beta} \mid \vec{\beta} \preceq \vec{\alpha} \text{ and } \vec{\beta} \text{ is non-conflicting in } w \text{ and } \vec{\beta} \text{ is } \preceq\text{-maximal}\}$$

Elements of $\texttt{Max}(\vec{\alpha}, w)$ are therefore joint actions non-conflicting in w where a minimal number of actions are inhibited. Notice that $\texttt{Max}(\vec{\alpha}, w) = \{\vec{\alpha}\}$ as soon as $\vec{\alpha}$ is non-conflicting in w.

Example 3. *The set* $\texttt{Max}(\langle \texttt{push}_a, \texttt{push}_b \rangle, w)$ *is:*
$\{\langle \texttt{wait}_a, \texttt{push}_b \rangle, \langle \texttt{push}_a, \texttt{wait}_b \rangle\}$.

Let us now define the concurrent update product, that implements the execution of joint actions.

Definition 3.6.3 (Concurrent update product)**.** *The* concurrent update product *of an epistemic model $\mathcal{M}$ and an action model $\mathcal{A}$ is the Kripke model $\mathcal{M} \boxplus \mathcal{A} = (W^{\boxplus}, (\approx_a^{\boxplus})_{a \in Agt}, \lambda^{\boxplus})$, where:*

- $W^{\boxplus} = \{(w, \vec{\beta}) \in W \times A^N \mid \vec{\beta} \in \texttt{Max}(\vec{\alpha}, u)$, $\vec{\alpha}$ *available in* $w\}$;
- $(w, \vec{\beta}) \approx_a^{\boxplus} (u, \vec{\gamma})$ *if* $w \approx_a u$ *and* $\vec{\beta}_b \approx_a^{\mathcal{A}} \vec{\gamma}_b$ *for all* b;
- $p \in \lambda^{\boxplus}(u, \vec{\beta})$ *if* $(\mathcal{M}, u) \models \bigwedge_{a \in Agt(\vec{\beta}, p)} post(\beta_a, p)$.

Let us now define the set of epistemic states that may result from

executing a joint action $\vec{\alpha}$ *in* $(\mathcal{M}, w)$*:*

$$(\mathcal{M}, w) \boxplus \vec{\alpha} := \{(\mathcal{M} \boxplus \mathcal{A}, (w, \vec{\beta})) \mid \vec{\beta} \in \mathtt{Max}(\vec{\alpha}, w)\}.$$

This set is a singleton when $\vec{\alpha}$ *is non-conflicting in* w*.*

Remark that by the component-wise definition of concurrent epistemic relations $\approx_a^{\boxplus}$ in Definition 3.6.3, the *identity* of who performs an action is common knowledge.

3.6.5 Concurrent Games

In this section, it is shown how DEL can represent infinite concurrent game arenas, in which players act simultaneously. It is then shown how, as for turn-based games (see sections 3.4 and 3.5), these infinite game arenas can in some cases be folded back into finite ones, when actions are all public, or when they are all propositional. Finally, it is shown that in these cases, one can model check the epistemic strategic logic $\mathsf{ATL}^*_{\mathrm{K}}$ on DEL-represented concurrent games. These results actually subsume those of previous sections, as turn-based games can be seen as particular cases of concurrent games, and $\mathsf{ATL}^*_{\mathrm{K}}$ captures distributed strategy synthesis, and much more.

Definition 3.6.4. *A concurrent DEL arena* $\mathfrak{G} = \langle \mathcal{M}, w_\iota, \mathcal{A} \rangle$*, or DEL arena from now on, consists of an initial pointed epistemic model* $(\mathcal{M}, w_\iota)$ *and an action model* $\mathcal{A}$ *that satisfy condition (H3') below.*

Observe that a concurrent DEL arena is thus the same thing as a turn-based DEL arena (Definition 3.5.1), except that it is no longer required hypothesis H1 and H2 from Section 3.5.1, which were used to model turn-based situations (and hence it is no longer needed variable turn too). However, it is still needed hypothesis H3 to ensure that agents know which actions are available to them.

Though the formal description of a concurrent DEL arena is very close to that of a turn-based DEL arena, the intended game is quite different.

The game starts in the initial epistemic state $(\mathcal{M}, w)$. In each round, each agent a chooses an action $\alpha \in A_a$ available in the current state $(\mathcal{M}', w')$, resulting in a joint action $\vec{\alpha}$ (it is assumed that each agent always has at least one available action; if needed, one may add a dummy action in the model). The next epistemic state is nondeterministically chosen by the scheduler among $(\mathcal{M}', w') \boxplus \vec{\alpha}$ (this choice is nontrivial if $\vec{\alpha}$ is conflicting), and the game goes on.

After n rounds in which the players chose joint actions $\vec{\alpha}_1, \ldots, \vec{\alpha}_n$, the epistemic state is of the form $\mathcal{MA}^n, (w, \vec{\beta}_1, \ldots, \vec{\beta}_n)$, where $\mathcal{MA}^n$ is defined by letting $\mathcal{MA}^0 = \mathcal{M}$ and $\mathcal{MA}^{n+1} = \mathcal{MA}^n \boxplus \mathcal{A}$, and each $\vec{\beta}_{k+1}$ is a maximal sub-action of $\vec{\alpha}_{k+1}$ consistent in $\mathcal{MA}^k, (w, \vec{\beta}_1, \ldots, \vec{\beta}_k)$. In the following, one may write $w\,\vec{\beta}_1 \ldots \vec{\beta}_n$ instead of $(w, \vec{\beta}_1, \ldots, \vec{\beta}_n)$, and call it a history. The *length* of a history in $\mathcal{MA}^n$ is defined as $|w\,\vec{\beta}_1 \ldots \vec{\beta}_n| = n$. Since the length of a history determines the epis-

temic model to which it belongs, one may omit it and write $w\,\vec{\beta}_1 \dots \vec{\beta}_n \models \phi$ instead of $\mathcal{MA}^n, w\,\vec{\beta}_1 \dots \vec{\beta}_n \models \phi$. It is required that players know their available actions:

(H3) **Players know their available actions:** For all histories of the form $w\,\vec{\beta}_1 \dots \vec{\beta}_n$ and $u\,\vec{\gamma}_1 \dots \vec{\gamma}_n$, if $w\,\vec{\beta}_1 \dots \vec{\beta}_n \approx_a u\,\vec{\gamma}_1 \dots \vec{\gamma}_n$, then the same actions are available to a in both worlds.

A concurrent DEL arena denotes a concurrent game arena, where positions are epistemic states attainable from the initial one, and moves are obtained by applying the concurrent update product of Definition 3.6.3. Because this product yields several epistemic states, this concurrent game arena is non-deterministic.

Definition 3.6.5. *Given a concurrent DEL arena $\mathfrak{G} = \langle \mathcal{M}, w_\iota, \mathcal{A} \rangle$, one can define the corresponding concurrent game arena $G_{\mathcal{M},\mathcal{A},W_\iota}$ as the tuple $(A, V, v_\iota, (Act_a)_{a\in Agt}, \Delta, (\approx_a)_{a\in Agt}, \lambda)$ where:*

- *A is the set of actions in $\mathcal{A}$,*
- *the set of positions is $V = \cup_{n\in\mathbb{N}} W^n$,*
- *the initial position is $v_\iota = w_\iota$,*
- *$Act_a(v) = \{\alpha \in A_a \mid v \models pre(\alpha)\}$,*
- *$\Delta = \{(v, \vec{\alpha}, v') \mid v' \in v \boxplus \vec{\alpha}\}$,*
- *$v \approx_a v'$ if $|v| = |v'| = n$ and $v \approx_a^n v'$, and*

- $\lambda(v) = \lambda^{|v|}(v)$.

Let us reason about strategic abilities and knowledge on infinite concurrent games given as concurrent DEL arenas. To do so, let us consider the logic $\mathsf{ATL}^*_{\mathrm{K}}$.

Definition 3.6.6. *The syntax of* $\mathsf{ATL}^*_{\mathrm{K}}$ *is given by the following grammar:*

$$\phi \ ::= \ p \ \mid \ \neg\phi \ \mid \ \phi_1 \vee \phi_2 \ \mid \ K_\phi \ \mid \ \langle C \rangle \psi$$
$$\psi \ ::= \ \phi \ \mid \ \neg\psi \ \mid \ \psi_1 \vee \psi_2 \ \mid \ X\psi \ \mid \ \psi_1 U \psi_2$$

with $p \in AP$ *and* $C \subseteq Agt$.

Formula $\langle C \rangle \psi$ reads as "coalition C has a strategy profile to ensure ψ", and the meaning of other operators is as usual. We let $\approx_C = \cup_{a \in C} \approx_a$, and define the semantics of $\mathsf{ATL}^*_{\mathrm{K}}$ as follows.

Definition 3.6.7. *Let* G *be a concurrent game arena,* v *a position and* π *a play. The semantics of* $\mathsf{ATL}^*_{\mathrm{K}}$ *is defined as follows, where* h

is a history in G, π is a play and $i \in \mathbb{N}$ is a point in time:

$$
\begin{array}{lll}
G, h \models p & \textit{if} & p \in \lambda(h) \\
G, h \models \neg\phi & \textit{if} & G, h \not\models \phi \\
G, h \models \phi_1 \vee \phi_2 & \textit{if} & G, h \models \phi_1 \textit{ or } G, h \models \phi_2 \\
G, h \models K_\phi & \textit{if} & \forall h' \in Hist^G \textit{ s.t.} h \approx_a h', G, h' \models \phi \\
G, h \models \langle C \rangle \psi & \textit{if} & \exists \sigma_C \in \Sigma_C \textit{ s.t.} \forall h' \approx_C h, \forall \pi \in Out(h', \sigma_C), \\
 & & G, \pi \models \psi \\
G, \pi \models \phi & \textit{if} & \pi[0] \models \phi \\
G, \pi \models \neg\psi & \textit{if} & G, \pi \not\models \psi \\
G, \pi \models \psi_1 \vee \psi_2 & \textit{if} & G, \pi \models \psi_1 \textit{ or } G, \pi \models \psi_2 \\
G, \pi \models X\psi & \textit{if} & G, \pi_{\geq 1} \models \psi \\
G, \pi \models \psi_1 U \psi_2 & \textit{if} & \exists i \geq 0 \textit{ s.t. } G, \pi_{\geq i} \models \psi_2 \textit{ and } \forall\, 0 \leq j < i, \\
 & & G, \pi_{\geq j} \models \psi_1
\end{array}
$$

Remark 5. *We use the uninformed semantics for the knowledge operators, which does not depend on the strategies used by the agents, and is the usual one used in epistemic planning (see [87, 90, 111] for more details). Also, we use the subjective semantics for strategic operators, in contrast with the objective one that only asks strategies to be winning from the actual current history (see [40]).*

Given a game arena G with initial position v_ι, let us write $G \models \phi$ if $G, v_\iota \models \phi$.

Let us study the following model-checking problem: given a DEL

game presentation $\mathfrak{G}$ and a formula $\phi \in \mathsf{ATL}^*_{\mathrm{K}}$, is it the case that $G_{\mathcal{M},\mathcal{A},W_\iota} \models \phi$? It is shown that this can be decided in two cases: when all actions are propositional, and when they all are public. In both cases, the proof goes via establishing that the generated DEL games can be finitely represented.

3.6.6 Decidability for public actions

A *deterministic* game arena with transition function δ is said to have *only public actions* if, for all positions v, v' and joint actions $\vec{\alpha}, \vec{\alpha}'$ such that $\vec{\alpha} \neq \vec{\alpha}'$, we have $\delta(v, \vec{\alpha}) \not\approx_a \delta(v', \vec{\alpha}')$. The following is known for $\mathsf{ATL}^*_{\mathrm{K}}$ on such arenas:

Theorem 3.6.1 ([26])**.** *Model checking* $\mathsf{ATL}^*_{\mathrm{K}}$ *on finite deterministic concurrent game structures with only public actions is* 2-Exptime*-complete.*

First, let us prove that DEL games with public actions can be finitely represented. The proof yields nondeterministic game arenas, that will be then transformed into deterministic ones that have only public actions in the sense of [26].

It will be needed the following result on the concurrent update product with public action models and a public scheduler (see Definition 3.6.2):

Lemma 3.6.1. *Let* $\mathcal{M}$ *be an epistemic model and* $\mathcal{A}$ *an event model. If the scheduler is public,* $\mathcal{A}$ *has only public actions and its ghost*

mapping $\circ : \mathsf{A}_\bullet \to \mathsf{A}_\circ$ *is injective, then for all worlds* $w, w' \in \mathcal{M}$*, all joint actions* $\vec{\alpha}, \vec{\alpha}'$*, if* $\vec{\alpha} \neq \vec{\alpha}'$ *then for all* $(w, \vec{\beta}) \in (\mathcal{M}, w) \boxplus \vec{\alpha}$ *and* $(w', \vec{\beta}') \in (\mathcal{M}, w') \boxplus \vec{\alpha}'$*, for all agents* a*,* $(w, \vec{\beta}) \not\approx_a (w', \vec{\beta}')$*.*

Proof. Assume that $\vec{\alpha}_b \neq \vec{\alpha}'_b$ for some $b \in Agt$. If agent b's action is kept in both $\vec{\alpha}$ and $\vec{\alpha}'$, i.e., $\vec{\beta}_b = \vec{\alpha}_b$ and $\vec{\beta}'_b = \vec{\alpha}'_b$, the result follows from the fact that $\vec{\alpha}_b$ and $\vec{\alpha}'_b$ are different public actions. If it is kept in one and inhibited in the other, say, $\vec{\beta}_b = \vec{\alpha}_b$ and $\vec{\beta}'_b = (\vec{\alpha}'_b)_\circ$, then again $\vec{\beta}_b$ and $\vec{\beta}'_b$ are public actions and they are different because $A = \mathsf{A}_\bullet \uplus \mathsf{A}_\circ$. Finally, if both are inhibited, i.e., $\vec{\beta}_b = (\vec{\alpha}_b)_\circ$ and $\vec{\beta}'_b = (\vec{\alpha}'_b)_\circ$, again these actions are different by injectivity of the ghost mapping. □

The injectivity assumption means that two different played actions cannot become the same after being inhibited, in which case the actions of DEL games could not be public in the sense of [26].

The proof that the infinite game arena $G_{\mathcal{M},\mathcal{A},W_\iota}$ induced by a DEL game $\mathfrak{G}$ with public actions can be finitely represented relies on the fact that updating an epistemic state with a joint action made of public actions can only decrease the size of the obtained epistemic states (when removing their disconnected components) as long as their ghosts are public too. It follows that only finitely many different epistemic models can be generated from a given initial state, up to isomorphism.

Proposition 3.6.1. *Given a DEL game* $\mathfrak{G} = \langle \mathcal{M}, w, \mathcal{A} \rangle$ *with only public actions, one can build a finite game arena* G *equivalent to* $G_{\mathcal{M},\mathcal{A},W_\iota}$ *and of size exponential in* $|\mathfrak{G}|$*.*

Because of the scheduler, the resulting arena is nondeterministic. To obtain Theorem 3.6.2 below, it essentially remains to show that one can transform these nondeterministic arenas into deterministic ones that have only public actions in the sense of [26].

Theorem 3.6.2. *Model checking* $\mathsf{ATL}^*_{\mathrm{K}}$ *on DEL concurrent games with public actions, public scheduler and injective ghost mapping is* 2-EXPTIME-*complete.*

3.6.7 Decidability for propositional actions + hierarchical information

A DEL game presentation presents *hierarchical information* if the set of agents Agt can be totally ordered $(a_1 < \ldots < a_N)$ so that $\approx_{a_i} \subseteq \approx_{a_{i+1}}$ for each $1 \leq i < N$, and similarly for concurrent game arenas. It is proved that when, in a DEL game presentation, all actions are propositional actions and information is hierarchical, model checking $\mathsf{ATL}^*_{\mathrm{K}}$ is decidable.

Let us first generalise to the concurrent setting a result from [91] which states that infinite turn-based DEL games induced by propositional models can be finitely represented. As for public actions, the game arena that we obtain is nondeterministic.

Proposition 3.6.2. *Let* $\mathfrak{G} = \langle \mathcal{M}, w_\iota, \mathcal{A} \rangle$ *be a DEL game where* $\mathcal{A}$ *is propositional. One can build a finite game arena* G *equivalent to* $G_{\mathcal{M},\mathcal{A},W_\iota}$*, of size exponential in* $|\mathfrak{G}|$*.*

Proof. Let AP_i be the atomic propositions involved. Let us define the concurrent game arena $G = (A, V, v_\iota, (Act_a)_{a \in Agt}, \Delta, (\approx'_a)_{a \in Agt}, \lambda')$, where:

- A is the set of events in $\mathcal{A}$,
- $V = W \cup A^N \times 2^{AP_i}$,
- $Act_a(w) = \{\alpha \in A_a \mid w \models pre(\alpha)\}$,
- $Act_a(\vec{\beta}, \nu) = \{\alpha \in A_a \mid \nu \models pre(\alpha)\}$,
- $(w, \vec{\alpha}, (\vec{\beta}, \nu)) \in \Delta$ if
  - $\vec{\alpha}$ is a valid joint action available in w,
  - $\vec{\beta}$ is a maximal consistent sub-action of $\vec{\alpha}$, and
  - ν is the new valuation after executing $\vec{\alpha}$ in w
- $((\vec{\beta}, \nu), \vec{\alpha}, (\vec{\beta}', \nu')) \in \Delta$ if
  - $\vec{\alpha}$ is a valid joint action available in ν,
  - $\vec{\beta}'$ is a maximal consistent sub-action of $\vec{\alpha}$, and
  - ν' is the new valuation after executing $\vec{\alpha}$ in ν
- $w \approx'_a w'$ if $w \approx_a w'$ in $\mathcal{M}$
- $(\vec{\beta}, \nu) \approx'_a (\vec{\beta}', \nu')$ if for all $b \in Agt$, $\vec{\beta}_b \approx_a \vec{\beta}'_b$ in $\mathcal{A}$
- $\lambda'(w) = \lambda(w)$ and $\lambda'(\vec{\beta}, \nu) = \nu$

It is not hard to check that the game obtained is equivalent to $G_{\mathcal{M},\mathcal{A},W_\iota}$. Note that the transition relation is well defined because all preconditions and postconditions are propositional, and thus it is enough to know the current valuation to evaluate them. □

Let us invoke the following result:

Theorem 3.6.3 ([31])**.** *Model checking* $\mathsf{ATL}^*_{\mathrm{K}}$ *with uninformed semantics is decidable on deterministic game arenas with hierarchical information.*

The result in [31] is for ATL^* without knowledge operators, and for the objective semantics of strategic operators. But their bottom-up algorithm can be easily extended to deal with knowledge operators and subjective semantics: one first performs a powerset construction on the game arena to include sufficient information to evaluate knowledge operators positionally (for instance using van der Meyden's *k-trees* [128]), and then one evaluates knowledge and strategic modalities in a bottom-up fashion.

Thanks to Proposition 3.6.2 and a reduction to deterministic arenas similar to the one presented in the proof of Theorem 3.6.2, we can use Theorem 3.6.3 and obtain that:

Theorem 3.6.4. *Model checking* $\mathsf{ATL}^*_{\mathrm{K}}$ *on DEL game presentations with propositional actions and hierarchical information is decidable.*

However the complexity is nonelementary, as it is already the case

for multiplayer reachability games with hierarchical information [106]. The number of exponentials will be the maximum between the number of agents with different observation power and the alternation of knowledge operators in the formula.

Chapter 4

A Comparative Study on the Most Common Model Checking Tool: MCMAS

4.1 Introduction

In formal methods for multi-agent systems, logics for the strategic reasoning have had a major role. Among the others, ATL* (and ATL) has come to the fore and largely explored for practical use [10]. More recently, Strategy Logic (SL)[93] has come out, where strategies are treated explicitly as first order objects and associated to agents by means of a binding operator. Let us recall that SL makes use of the binding operator and strategy quantifiers along its syntax. For the

latter, we have the existential operator $\exists x$ and the dual universal operator $\forall x$ that can be read as "for some strategy x, ..." and "for all strategies x, ...", respectively. The binding operator (x, a) means that "by using strategy x, agent a can achieve...". SL is much more expressive than ATL*, and able to express important solution concepts among which the Nash Equilibrium. The high expressiveness of SL comes at a price: the model-checking problem is non-elementary. This has led at looking for meaningful elementary fragments of SL, among the others SL[1G] [94] and SL[SG] [27].

SL[1G] refers to SL formulas of the form $\wp\flat\phi$ where $\wp$ is a quantification prefix on strategies, $\flat$ a binding, and ϕ an LTL formula. SL[1G] subsumes ATL* and shares with it important features, among which a 2ExpTime solution for the model checking problem [94] (implemented in MCMAS [42]). SL[SG] further restrict SL[1G] formulas in a way similarly as ATL restricts ATL*. Thus, SL[SG] can be seen as the extension of ATL to arbitrary quantification on the agents' strategies [27]. Interestingly, SL[SG] can express meaningful concepts such as the Stakelberg equilibrium and coercion in voting, while providing a polynomial-time solution for the model checking problem. Notably, no direct implementation for SL[SG] in MCMAS has been exploited yet (since SL[1G] subsumes SL[SG], clearly MCMAS can handle SL[SG] formulas). In the sequel, we give some evidence that such an implementation should be played out, instead. Indeed, by restricting to SL[SG]

formulas that can be translated to ATL, it can be shown that over such formulas the standard MCMAS implementation for ATL works much faster than the one implemented for SL[1G].

4.2 Background on SL and MCMAS

In this section, some basic notions about SL[SG] and the main features of the most largely employed tool for model checking, MCMAS, are reported.

4.2.1 Strategy Logic with Simple Goals

Let us briefly recall the syntax, and the main results for SL[SG] ([27]). Let AP, Agt, and Var be the sets of atomic propositions, agents, and variables, respectively. Given $A \subseteq Agt$ and $V \subseteq Var$, we define a *binding prefix* as a finite sequence $\flat \in \{(x, a) \mid a \in A \text{ and } x \in V\}^{|A|}$ such that $|\flat| = |A|$, and every agent $a \in A$ occurring exactly once in $\flat$. Contrarily, the same variable $x \in V$ can occur several times in $\flat$, allowing agents in A to use the same strategy more than once. A *quantification prefix* over the set V of variables is a finite sequence $\wp \in \{\exists x, \forall x \mid x \in V\}^{|V|}$ such that $|\wp| = |V|$ and every variable $x \in V$ occurs exactly once in $\wp$. $Qnt(V) \subset \{\exists x, \forall x \mid x \in V\}^{|V|}$ and $Bnd(A) \subset \{(x, a) \mid a \in A \text{ and } x \in Var\}^{|A|}$ denote the sets of all quantification and binding prefixes over variables in V and agents in

A, respectively.

Definition 4.2.1 (Syntax of SL[SG]). *Assuming the notion of free variable as reported in [94], given a formula ϕ in* SL[SG], $\flat \in Bnd(Agt)$, $\wp \in Qnt(free(\flat\phi))$, *and $p \in AP$, ϕ can be expressed as follows:*

$$\phi \ ::= \ p \mid \neg\phi \mid \phi \wedge \phi \mid \wp\flat X\phi \mid \wp\flat(\phi U \phi)$$

Where X and U are the temporal next and until operators, respectively.

Let us conclude this section by recalling two main results for SL[SG].

Theorem 4.2.1 (Expressiveness of SL[SG] [27]). *Strategy Logic with Simple Goals has strictly greater expressive and distinguishing power than* ATL.

Theorem 4.2.2 (Complexity of SL[SG] [27]). *The model checking for Strategy Logic with Simple Goals is* PTIME*-complete.*

4.2.2 MCMAS

Let us recall that MCMAS is a tool to model check formulas over multi-agent systems. The setting is typically modeled through interpreted system, using a dedicated language to formalize it, ISPL (Interpreted System Programming Language), and it involves two types of agents: the *standard* agent and the environment one. The latter is used to describe boundary conditions and variables shared among the *standard*

agents. The model checking procedure is based on binary decision diagrams (BDD, for short), commonly used to represent boolean functions in a compact manner. They consist of finite directed acyclic graphs with a unique initial node, in which each internal node is a boolean variable, and each terminal node is a truth value. The final aim is to determine, given an interpreted system $\mathcal{I}$, an initial state g, and a formula ϕ, if: $\mathcal{I}, g \models \phi$.

4.3 Comparison of Existing Tools

The tool which is commonly used to model check multi-agent systems is MCMAS, which takes as input a MAS specification and a set of formulas to be verified. Initially, MCMAS was designed to solve formulas expressed in CTL and ATL. Later, with the expansion of Strategy Logic, and in particular with SL[1G], a new release of the model checker has been implemented, which supports SL[1G] formulas.

The SL[1G] version of MCMAS has been tested against the standard ATL version of MCMAS, over SL[SG] formulas which can be translated in ATL. Notice that formulas written in SL[SG] are clearly supported by the SL[1G] version of MCMAS, since SL[SG] $\subseteq$ SL[1G]. The idea was to understand if such version of MCMAS behaves well even on the SL[SG] fragment, or if it would be the case of defining a new version for it.

The study consists in testing the two versions of the model checker

from two points of view:

1. By scaling on the number of variables;

2. By scaling on the number of agents.

	ATL-MCMAS		SL[1G]-MCMAS	
shells	**time**	**space**	**time**	**space**
200	73,37	64M	2,03	30M
400	248,87	194M	8,327	45M
600	1263,63	410M	214,47	71M
800	1697,09	708M	230,26	109M
1000	613,094	169M	5630,94	134M
1200	8031,69	1660M	1473,87	239M
1400	9992,47	1947M	4732,29	283M

Table 4.1: Results of multiple executions on the shells example by varying the number of possible shells: comparison between the standard ATL MCMAS and the SL[1G] extension.

In Table 4.1, the behavior of MCMAS in the standard version for ATL against the SL[1G] one is shown, by varying the number of available variables in the *shell game*, in which notice that the player has to guess which shell hides an object. In the setting, the players are the environment, that places the object underneath the selected shell, and the guesser who has to guess one among the available ones. The simulation starts from 200 possible shells: the behavior of SL[1G] of MCMAS is better than the one of the standard ATL version, both in terms of time and space, by assigning a truth value to the same set of formulas in a one-magnitude-smaller time, and half of the space.

Moving to 400 shells, the time needed for SL[1G] MCMAS is two magnitude smaller, and the space is 4 times less. If we give 600 shells, the difference is still high, again with a time of one magnitude smaller, and the needed space 5 times less. The trend is still the same with 800 available shells. Instead, when we reach 1000 shells, MCMAS for ATL seems to find an efficient BDD technique to solve the model checking, but the results are not confirmed by any other test with different input. Indeed, with 1200 and 1400 shells, MCMAS for SL[1G] keeps being better: even if the time results are of the same magnitude as the ATL MCMAS corresponding ones, the space needed is approximately 6 times less in both cases. In Table 4.2, instead, it is shown how the model checkers' response changes by varying the number of agents in the *voting game* [68]. Let us recall that, in the most simple scenario, this game involves two players: a voter and a coercer. After voting, the voter can decide to hand in proof of his vote, or not to do it. In the same way, the coercer can decide to punish the voter, or not to do it. In our setting, the coercer is modeled through the environment, while we tested the tool by increasing the number of voters. Precisely, by moving from 4 to 12 voters, the standard ATL-MCMAS behavior does not change significantly, by providing the truth value of the input formula in times and spaces that are of the same magnitude. On the contrary, with the SL[1G] extension of MCMAS both the time and the space needed for the execution are increased of one magnitude when

	ATL-MCMAS		SL[1G]-MCMAS	
voters	**time**	**space**	**time**	**space**
4	0,035	9M	0,29	20M
6	0,038	9M	18,42	697M
8	0,027	9M	**aborted**	
10	0,09	10M		
12	0,08	10M		

Table 4.2: Results of multiple executions on the voting example by varying the number of agents involved: comparison between the standard ATL MCMAS and the SL[1G] extension.

the number of voters increases from 4 to 6. Instead, if we simply move it to 8 voters, the model checker forces its irregular termination, without assigning any truth value to the input formulas.

Let us provide some examples of formulas used to obtain the results just shown. For the *shell game* with 200 shells, we run the SL[1G] version of MCMAS over the following SL[SG] formula:

$$\varphi_s = \forall e \, \exists g \, (e, Environment)(g, Guesser) F \; win$$

requiring that, no matter what the strategy of the environment is, there always exists a strategy for the guesser to finally win. Then, the standard ATL version of MCMAS run on the corresponding ATL formula:

$$\psi_s = \langle\langle g \rangle\rangle F \; win$$

where g is a coalition made by the guesser alone.

For the *voting scenario* with 4 voters, instead, the SL[SG] formula

which is tested is the following:

$$\varphi_v = \wp\, \flat\, F((vote_1^1 \rightarrow\ punish_1)\ and\ (vote_1^2 \rightarrow\ punish_2)\ and\ (vote_1^3 \rightarrow\ punish_3)\ and\ (vote_1^4 \rightarrow\ punish_4))$$

where:

- $\wp = \forall v_1\, \forall v_2\, \forall v_3\, \forall v_4\, \exists e$ and
- $\flat = (v_1, Voter_1) \ldots (v_4, Voter_4)(e, Environment)$.

The above specifies that, no matter what the strategies of the voters are, there always exists a strategy for the environment such that if a voter votes the candidate 1 finally he will be punished. The corresponding ATL formula is:

$$[\![g_v]\!]F(\langle\langle g_e\rangle\rangle F(((vote_1^1 \rightarrow\ punish_1)\ and\ (vote_1^2 \rightarrow\ punish_2)\ and\ (vote_1^3 \rightarrow\ punish_3)\ and\ (vote_1^4 \rightarrow\ punish_4))$$

where g_v is the coalition of voters and g_e is the environment alone.

To develop a tool for SL[SG] model checking, one would expect to modify the existing ATL MCMAS in a way that makes it able to solve SL[SG] formulas. It will be needed to modify opportunely the parser to accept SL[SG] syntax, but one would also have to integrate the new model checking algorithm in the existent tool, to reflect the SL[SG] semantics.

Part IV

Part Three

Chapter 5

A Nash-Equilibrium-Based Parking Algorithm

5.1 Introduction

The growth of Artificial Intelligence applications to automotive is constantly increasing the request for smart solutions to parking. This research field is well identified as *smart parking* (see [76]).

The competitive nature of the parking process, during which the drivers compete in order to get an available parking slot for their cars, is the inspiration of the proposed solution. Indeed, by exploiting basic settings of the strategic reasoning for multi-agent systems, it has been

modeled the parking process as a competitive multi-player game in which each car is an agent interacting with all the other ones, with the ultimate goal of getting an available slot that satisfies its own constraints. The *parking problem* to face concerns parking as many cars as possible, while satisfying their requirements.

A multi-agent system is made of autonomous entities, with distributed information, computational capabilities, and possibly diverging interests. These kind of systems have been exploited in several fields: electronic [120] and industrial process control [17], economy [102], home automation [142], open system verification [11], to name a few.

The way autonomous agents can interact with each other can be classified into two categories: competitive and cooperative. In the former case, there is no a-priori agreement among agents, as they try to maximize their own objective, no matter what the objectives of the other agents are. In the latter case, the agents coordinate among them in order to get the best outcome possible for everyone [136].

The contribution of this proposal [41] is twofold. From one side, it comes up with an effective multi-agent game model for the parking problem considered. From the other side, it is provided an algorithm (and its implementation in a tool), working in quadratic time, that allows a fair allocations of the parching slots by satisfying a Nash equilibrium. As will be proved later on practical scenarios, this is a valuable compromise with respect to an optimal, but exponential,

brute-force solution that would check all possible distribution of cars over available slots.

Also, as expected, the solution is better than any greedy FIFO approach. Indeed, consider a scenario in which there are three vehicles, V_1, V_2, and V_3, looking for a parking, and three slots available A, B, and C. Assume now that V_1, V_2, and V_3 have up to 7, 5, and 3 minutes to accomplish the parking, respectively. Also, assume that slots A, B, and C require 2, 3, and 5 minutes to be reached, respectively. Assume now that V_1 picks A and V_2 picks B; then, V_3 would not have enough time to reach the remaining slot C. Contrarily, a solution that allows parking all vehicles by accommodating their requirements is to assign V_1, V_2, and V_3 to C, B, and A, respectively. This is exactly what the proposed algorithm would return as a solution.

5.2 Related Work

As explained before, several algorithmic solutions have been proposed in the VANET research field for the parking problem. Less common is the application of Artificial Intelligence and game-theoretic approaches to the solution of the parking problem. A work that I like to mention, along the second line, is [70], which is probably the closest to what is going to be presented. In that article, the authors also propose a parking solution based on the Nash equilibrium. However, their approach is quite different from the one used here. First, they provide

a numerical solution, thus, they do not give any algorithm nor any tool to solve the problem. Second and more importantly, they consider a scenario with both private and public parking slots, and the drivers' payoffs strongly rely on such a topology; it is not clear to us whether and how to massage their model to accommodate our system.

Many other smart parking mechanisms have been proposed based on a multi-agent game setting. In [83], the authors simulate the drivers' behavior through a multi-agent system, by modeling the environment on the basis of cellular automata. In [29], the authors chose two different agents, the user and the administrator, and manage the communication through three layers. They focus on the architecture rather than the model setting and the strategic reasoning. Similarly, authors of [69] provide an E-parking system, based on multi-agent systems aimed to optimize several users' preferences. Moreover, in [99] authors manage the parking problem with a cooperative multi-agent system, by relying on a priority mechanism. In [104], authors also focus on equilibrium notion, but they study the Rosenthal equilibrium rather than the Nash one, which describes a probabilistic choice model. Finally, in [81] the authors also consider a Nash Equilibrium applied to cars, but they only used it to talk about traffic, rather than parking.

5.3 A Real Scenario

As case study, it has been selected the parking area of the *Federico II Hospital Company* in Naples, one of the biggest and most specialized hospital in the South of Italy, whose construction goes back to the early Sixties.

The Hospital, as it is schematically depicted in Figure 5.1, is made of 21 building blocks, distributed over $440.000m^2$, and provides in total one thousand of beds for ordinary recovery and two hundreds of beds for day-hospital use. The parking space, having 2684 slots in total, consists of 21 independent areas, and is mainly used by patients and, in turn, by the 3400 employees (doctors, nurses, technicians, administrators, etc.).

The hospital has four guarded gates, one of which is for pedestrian. The car gates are preceded by a road where cars line up for the necessary checks. In average, it is estimated that there are 4.600 car accesses per day. There is no policy about the allocation of the parking places and, except for few reserved ones, each driver chooses by its own the slot. This disorganized solution produces a huge traffic congestion, bottlenecks at the entrance, and an unbalanced distribution of the cars over the parking area. More importantly, it does not take into account the specific constraints and some physical limitations of the users, such as walking issues or urgency. In the most crowded hours, in average, the drivers spend more than 20 minutes to find a parking

slot or, even worst, they leave the parking area by missing available slots.

In order to efficiently apply our tool, let us assume that the list of available slots in every area of the hospital is known at runtime. Also, let us make use of all information the car passengers have to pass to the hospital before entering, and in particular their logistics. Finally, let us assume that the drivers will be followed while driving inside the parking area, by means of tracking devices (GPS, smartphone, videocameras, etc.).

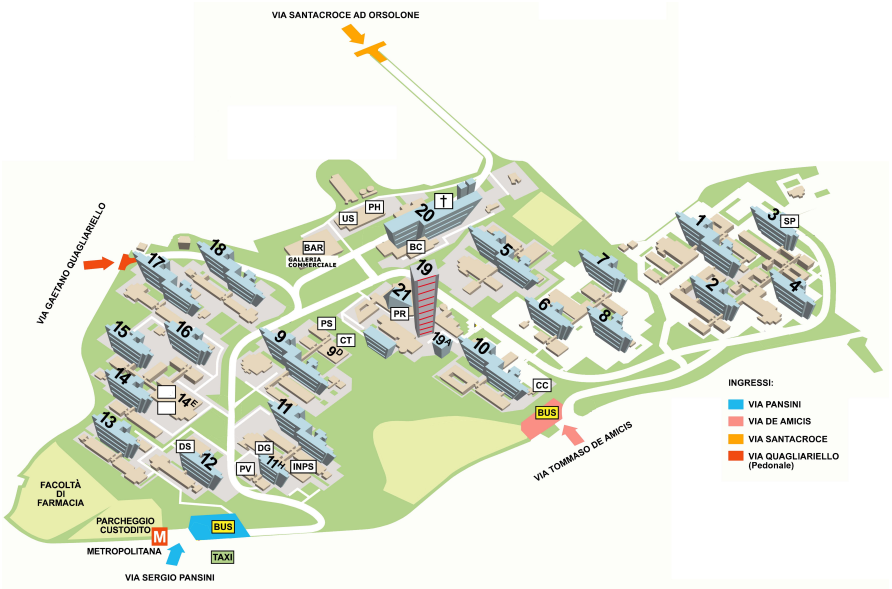


Figure 5.1: Graphic representation of the A.O.U. Federico II

Having all this information at his disposal, the tool works as follows: it takes all cars in queue on the roads in front the car gates, as well as all the specific needs and constraints of their occupants. Then, it processes the data and following the algorithm described in the sequel, it opportunely associates the available slots to the cars. In

particular, the tool will access both the Employers Data Center and the Online Booking Center of the hospital and, thanks to the latter, the tool will know which kind of services the patients need, date and time of their appointments, possible walking limitations and handicaps, etc. Finally, note that the tool operates in stages, processing one bunch of cars at the time, as they are in queue. Someone may criticize this solution and propose an offline allocation instead. It has been decided not to follow this solution for two main reasons: first, the hospital is highly dynamic in slot requests and, more importantly, slots are very limited in numbers, so it is better to allocate slots only when cars show up.

5.4 Parking Game Structure

In this section, the *Parking Game Structure* model, (*PGS*, for short), is introduced. It is the game model that will be used along our algorithm to reasoning about the parking problem to address. The model definition takes inspiration from the scenario described previously. Thus, in a PGS, the players are cars with their needs and constraints. Also, the PGS takes into account all the specification about the slots, in particular their location, their availability, the time they require to be reached from the entrance, and so on.

Formally the *Parking Game Structure* is defined as follows:

Definition 5.4.1 (Parking Game Structure)**.** *The Parking Game Structure (PGS) is a tuple:*

$$\mathcal{G} = (Agt, S, G, g, F, T, R) \tag{5.1}$$

where:

- $Agt = \{a_1, \ldots, a_n\}$ *is a set of* agents*, i.e., the cars,*
- $S = \{s_1, \ldots, s_m\}$ *is a set of parking* slots*,*
- $F = \{f_1, \ldots, f_n \mid f_i \in [0, 1]\}$ *is a set of* resilience *values,*
- $G = \{g_1, \ldots, g_l\}$ *is a set of* gates*,*
- $g : Agt \to G$ *is a function associating agents to gates,*
- $AT = \{t_1, \ldots, t_n\}$ *is a set of* agent-time *values, where* t_i *is the time limit the car* a_i *has for parking,*
- $RT = \{r_{(1,1)}, \ldots, r_{(m,l)}\}$ *is a set of* reaching-time *values, where* $r_{(i,j)}$ *is the time needed to reach the parking slot* s_i *from gate* g_j*.*

Regarding the set of resilience indexes F, note that each f_i is associated with agent a_i and it has a twofold use: first, it imposes an order among agents; second, it affects the final pre-emption order. This will be more clear below. For simplicity, we assume that all the resilience indexes are different, i.e., $f_i \neq f_j$ for every $1 \leq i < j \leq n$. The indexes in F can be set manually as input, however it is important to report

that, for the case study introduced previously, the values have been obtained automatically by processing the information coming from the Employers Data Center and the Online Booking Center of the hospital; in particular, for the patients, the resilience index represents their movement ability, therefore, the lower the rate, the more favored the patient.

A *strategy* for an agent a_i consists of choosing a slot $s_j \in S$. Formally it is a function $Str : Agt \to S$. A *strategy profile* is an n-uple $\overline{s} = (\overline{s_1}, \ldots, \overline{s_n})$ of strategies, one for each player. Formally, in $\overline{s}$, for each i, we have $Str(a_i) = \overline{s_i}$. It is worth noting that it may happen that two or more players choose the same strategy. Next, it will be defined the *costs associated* to $\overline{s}$ as a tuple of costs $\overline{c} = (\overline{c_1}, \ldots, \overline{c_n})$. Then, a *payoff* π of a strategy $\overline{s}$ is defined as a sum of all such $\overline{c_i}$, i.e., $\pi(\overline{s}) = \sum_i \overline{c_i}$, and by π_i we denote the $i-$th cost value of that tuple.

Definition 5.4.2. *Let $a_i \in Agt$ be an agent with $g(a_i) = h$ and $\overline{s} = (\overline{s_1}, \ldots, \overline{s_n})$ be a strategy profile, with $\overline{s_i} = s_j$ for an $s_j \in S$. One can define the* costs associated *to $\overline{s}$ as the tuple $\overline{c} = (\overline{c_1}, \ldots, \overline{c_n})$ where*

each $\overline{c_i} =$

$$\begin{cases} f_i \cdot (t_i - r_{(j,h)}) & \textit{if 1. } (t_i - r_{(j,h)}) \geq 0, \textit{ and} \\ & \textit{2. there is no } a_{k \neq i} \textit{ such that} \\ & f_k < f_i, \ \overline{s_k} = \overline{s_i}, \ \textit{and} \\ & (t_k - r_{(j,p)}) \geq 0, \textit{ with } g(a_k) = p \\ \infty & \textit{otherwise} \end{cases} \tag{5.2}$$

In words, the value $\overline{c_i}$ is a finite value if the agent a_i has enough time to reach the parking slot s_j and such a slot has not been taken from any other agent a_k with a lower resilience (i. e., $f_k < f_i$). Then, the value, when it is finite, reflects how much time it is left to the agent after he has reached the assigned slot (with respect the total time he has at his disposal). Conversely, the infinity value corresponds to the worst possible outcome for the agent a_i, which reflects the fact that he cannot park at the slot s_j. At this point, it should be intuitive that the problem of looking for an optimal strategy profile $\overline{s}$ can be reduced to the problem of minimize[1] the corresponding vector of associated costs $\overline{c}$. Unfortunately, this is in general not an easy task. In particular, a brute-force algorithm checking all the possible strategy profiles is unfeasible as it requires exponential-time. Conversely, one can suggest adopting a Nash equilibrium solution which provides, by definition,

[1]Note that the minimization guarantees that the best slots are kept for future use, so to focus on the continue allocation process rather than the single stage. At any rate, one can use maximization (in a finite domain) without affecting the rest of the algorithm.

a satisfactory solution and, along with our setting, it just requires quadratic-time. In the sequel, such a solution is going to be presented. Also, it will be provided a solution based on a greedy behavior of the players, which reflects the current behaviour of drivers at the parking of the hospital described: each car takes the first available parking slot which satisfies its needs. By means of a toy example, it is possible to show that the solution based on the Nash equilibrium over-perform the one based on the greedy behaviour of the players.

5.5 The Parking Slot Selection Game

In this section, first a toy example is provided, then the *Parking Slot Selection Game* (PSSG, for short) is introduced, and a solution by means of a Nash equilibrium calculation is proposed. There is also comment on the greedy approach and a comparison with the provided solution. For a matter of precision, let us recall the notion of Nash equilibrium.

W.l.o.g., in the sequel the model is restricted to parking structures having just one gate. This means that one can get rid of g and G in PGS as given in Definition 5.4.1, as well as the second index of the reaching-time values in RT. This also simplifies the definition of costs associated with strategy profiles.

5.5.1 A Running Example

Let us consider a parking place with 3 slots available and 3 cars aiming at parking.

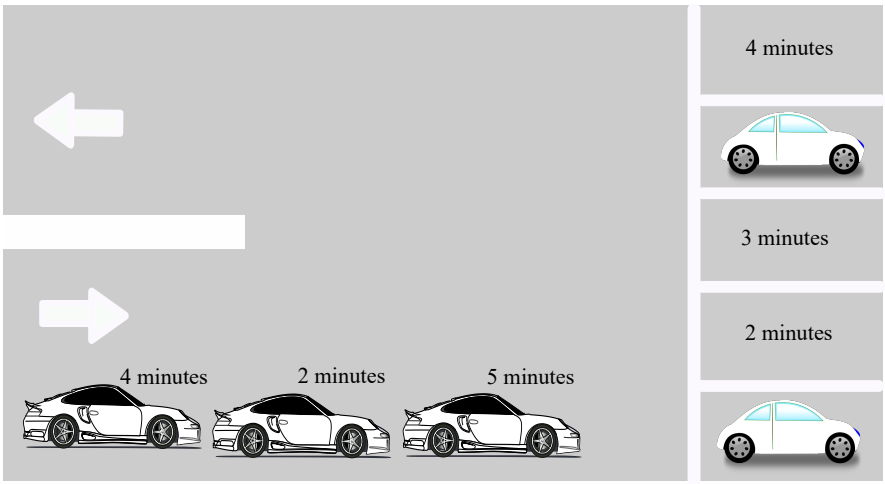


Figure 5.2: 3-players-3-slots Game.

Let us suppose that the first, the second, and the third car have respectively 5, 2, and 4 minutes available to park and that, as associated resilience they have 0.5, 0.1, and 0.009, respectively. Also, suppose that the first, the second, and the third slot require 2, 3, and 4 minutes to be reached, respectively. We call such a game the 3-players-3-slots game and it is reported in Figure 5.2.

5.5.2 A Greedy Solution

When a car is approaching to the parking, a greedy solution is to occupy the first slot it can get. This approach leaves to the car a free will to park in the slot that best fits its constraints, without paying attention to the other cars requirements. This easy-to-design solution

may lead to a non optimal vehicles allocation, as it may leave out some cars (not able to park), as the remaining slots may not satisfy their requirements.

To give an example, let us consider the scenario described in Section 5.5.1. In this situation, the first car would choose the closest slot (the one that requires 2 minutes to be reached). Then, the second car would not be able to park, because all the remaining free slots are too expensive in terms of time. For this reason, it has been looked for a better solution that would exploit the car parking potentialities at the best by means of a satisfactory distribution of slots among cars, and that would be computationally easy to be calculated on the fly.

5.5.3 Nash Equilibrium Based Solution

In game theory, a well-conceived solution concept that ensures a robust form of satisfaction among players is *Nash equilibrium*. This concept was deeply investigated and well formalized by John Nash in the fifties, both under pure and mixed strategies (see [126] for more details). In the basic definition, it is said that in a multi-player game, all players, moving concurrently, reach a Nash equilibrium if none of them has the incentive to unilaterally deviate from that equilibrium. By casting this in our parking scenario, one can try to reach a situation in which all drivers are associated to parking slots, by means of an equilibrium over their constraints. In other words, our goal is to provide a strategic

profile (parking slot assignment) in which no player wants to change his slot unless some other players want to change theirs.

Following the model definition given in Definition 5.4.1 and the observations made above, it is possible to formally introduce the *Parking Slot Selection Game* to address, as follows.

Definition 5.5.1 (Parking Slot Selection Game)**.** *The Parking Slot Selection Game (PSSG) has an input and an output defined as follows:*

- *Input: a PGS $\mathcal{G}$, as given in Definition 5.4.1.*
- *Output: a strategic profile $(\overline{s}_1^*, \ldots, \overline{s}_n^*)$ providing a Nash equilibrium for $\mathcal{G}$. A strategy profile $(\overline{s}_1^*, \ldots, \overline{s}_n^*)$ is a Nash equilibrium iff $\forall\, \overline{s}_1, \ldots, \overline{s}_n \in S$ it holds that:*

$$\begin{aligned}
&\pi_1(\overline{s}_1^*, \ldots, \overline{s}_n^*) \leq \pi_1(\overline{s_1}, \overline{s}_2^* \ldots, \overline{s}_n^*)\\
&\pi_2(\overline{s}_1^*, \ldots, \overline{s}_n^*) \leq \pi_1(\overline{s}_1^*, \overline{s_2} \ldots, \overline{s}_n^*)\\
&\vdots\\
&\pi_n(\overline{s}_1^*, \ldots, \overline{s}_n^*) \leq \pi_1(\overline{s}_1^*, \overline{s}_2^* \ldots, \overline{s_n})
\end{aligned}$$

In words, the PSSG consists in looking for a strategy profile that, with respect to the associated costs, no players has an incentive to unilaterally change his choice.

Similarly to the PSGG, one can define the *Greedy Parking Game* (*GPG*, for short). To give some details, first assume that in an GPG

players are ordered, then the strategy profile $(\overline{s}_1^*, \ldots, \overline{s}_n^*)$ is such that for each agent a_i, it holds that $\overline{s}_i^*$ is the best choice (in terms of minutes to reach it) over $S \setminus \{\overline{s}_1^*, \ldots, \overline{s}_{i-1}^*\}$.

5.5.4 A Solution to the 3-players-3-slots Game

Let us consider again the 3-players-3-slots game described in Section 5.5.1. It us now shown a solution based on the satisfaction of a Nash equilibrium. As we will see in a while, such a solution allows accommodating all cars, while satisfying all their constraints, contrarily to what we have seen with the greedy solution. Later, it will be shown that this is true in general and not just for the case of our specific example. Let us formally describe the 3-players-3-slots example by means of a PGS $\mathcal{G}_3$ whose components are defined as follows:

- $Agt = \{car_1, car_2, car_3\}$ is the set of cars,
- $S = \{slot_1, slot_2, slot_3\}$ is the set of parking slots,
- $AT = \{5, 2, 4\}$ is the set of time-values car_1, car_2, and car_3 have at at their disposal, respectively,
- $RT = \{2, 3, 4\}$ is the set of times needed to reach the slots $slot_1$, $slot_2$, and $slot_3$, respectively,
- $F = \{0.5, 0.1, 0.009\}$ is the set of cars resilient values,

- The cost function is reported in Table 5.1, in the last three rows. For instance, the triple $(\infty, \infty, 0.018)$ represents the case in which all cars decide to park in the same slot $slot_1$; so, car_3, which has the lowest resilience value, gets it at a cost of 0.018 (i.e., $(4-2) \cdot 0.009$), while the other cars leave the process incomplete, as they get ∞.

By a matter of calculation, one can check that there exists only one Nash equilibrium, which corresponds to $\overline{s} = (slot_2, slot_1, slot_3)$, with $\overline{c} = (1, 0, 0)$ (in bold in Table 5.1), and $\pi(\overline{s}) = 1$.

5.5.5 A Solution for the PSSG

In this section, it is introduced the algorithm for the solution to the problem described in Definition 5.4.1. First, the pseudo-code in Algorithm 8 is provided, then it is described how it works and report on its time complexity.

With the first iteration, the car with the lowest resilience index, *actualCar*, is selected from the queue, through the function *priorityCar(·)*, which takes as input the set of cars and returns the one with the lowest resilience index respect to the others. The variable cost *outcome* is associated an infinity value, the worst possible one. In the second iteration, the algorithm computes the costs resulting from the function *c(·)*, which takes as input a car and a slot. The value of the *outcome* is updated with the value of the best cost computed. Among the avail-

		car_3											
		$slot_1$			$slot_2$			$slot_3$					
		car_2			car_2			car_2					
		$slot_1$	$slot_2$	$slot_3$	$slot_1$	$slot_2$	$slot_3$	$slot_1$	$slot_2$	$slot_3$			
car_1	$slot_1$	$\infty, \infty, 0.018$	$\infty, \infty, 0.018$	$\infty, \infty, 0.018$	$\infty, 0, 0.009$	$1.5, \infty, 0.009$	$1.5, \infty, 0.009$	$\infty, 0, 0$	$1.5, \infty, 0$	$1.5, \infty, 0$			
	$slot_2$	$1, \infty, 0.018$	$1, \infty, 0.018$	$1, \infty, 0.018$	$\infty, 0, 0.009$	$\infty, \infty, 0.009$	$\infty, \infty, 0.009$	**1,0,0**	$1, \infty, 0$	$1, \infty, 0$			
	$slot_3$	$0.5, \infty, 0.018$	$0.5, \infty, 0.018$	$0.5, \infty, 0.018$	$0.5, 0, 0.009$	$0.5, \infty, 0.009$	$0.5, \infty, 0.009$	$\infty, 0, 0$	$\infty, \infty, 0$	$\infty, \infty, 0$			

Table 5.1: Cost function values for 3-drivers-3-slots instance of the game.

Algorithm 8 Algorithm for the solution of the PSSG.

Input: Queue of ready vehicles
Output: Slot allocation

```
1: while carQueue ≠ null do
2:     actualCar = priorityCar(carQueue).
3:     outcome = ∞.
4:     for slot ∈ setAvailableSlots do
5:         po = c(actualCar, slot).
6:         if po ≥ 0 & po < outcome then
7:             outcome = po.
8:             assignSlot(actualCar, slot)
9:             setNotAvailable(slot).
10:        end if
11:    end for
12: end while
```

able slots, the one with the best result is assigned to the *actualCar*. Once assigned, the slot is remove from the set of the available ones, with the function *setNotAvailable(·)*.

Theorem 5.5.1 (Correctness of Algorithm 8)**.** *Algorithm 8 computes the Nash equilibrium for the game.*

Proof (Sketch). Proving that the algorithm provides a Nash equilibrium is quite trivial. Assume by contradiction that $\overline{s} = (\overline{s_1}, .., \overline{s_n})$ is the solution provided from our algorithm and it is not a Nash equilibrium. Then, by definition of Nash equilibrium, there must exist an agent, let us say agent a_i, whose strategy s_j is not the best, while fixed the strategies for the other players. Hence, there exists another strategy s'_j for the agent a_i, such that the payoff of s'_j is better than the one for s_j (given the same strategies for the other players). But

if such a strategy s'_j exists, then it would be found at row 6 of our algorithm, and it would be chosen as the final strategy for agent a_i. But this clearly contradicts the hypothesis that $\overline{s} = (\overline{s_1}, .., \overline{s_n})$ is the solution provided. □

Theorem 5.5.2 (Complexity of Algorithm 8)**.** *The complexity of Algorithm 8 is quadratic with respect to the number of agents involved in the game, in the worst case.*

Proof. Let us take into account the worst possible scenario, by considering the case in which no vehicle obtains a parking slot, and let us compute $\mathcal{C}(PSSG)$ as the complexity of the Parking Slot Selection Game.

The proof proceeds by analyzing the complexity of the most expensive operations, from the inner ones to the outer ones. It will be used the notation $\mathcal{C}(r)$ to indicate the complexity of the code from the r-th row of the Algorithm 8.

The function $assignSlot(Car, slot)$ performs simple assignments, with a constant complexity $\mathcal{C}(7) = O(1)$.

The inner loop does not perform any slot assignment, in the considered worst case, since none of them satisfies the constraints of the cars to be allocated. Hence, the inner loop is repeated as many times as $|S|$, where S is the set of slots, according to Definition 5.4.1. Assuming that $|S| = m$, we can deduce that $\mathcal{C}(3) = \sum_{i=1}^{m} \mathcal{C}(7) = \sum_{i=1}^{m} O(1) = O(m)$.

The outer loop is performed as many times as the number of cars,

i.e, the agents. Since by Definition 5.4.1 $|Agt| = n$, we obtain that $\mathcal{C}(1) = \sum_{j=1}^{n} \mathcal{C}(3) = \sum_{j=1}^{n} O(k) = O(nm)$.

Assuming that, in the worst case, n and m are of the same order, we can conclude that the total complexity is $\mathcal{C}(PSSG) = O(n^2)$. □

5.6 Evaluation

In this section, we compare the performances between executing the greedy solution to solve GPGs and Algorithm 8 to solve PSGGs. We have run 10 times the two approaches on a growing number of cars and slots. All values and time-limit needed have been generated randomly. Results have been collected in Table 5.2. Each column represents a different execution of the two approaches with the corresponding input parameters, while the rows keep track of the two analyzed solutions. Each entry contains the number of cars that have been able to park successfully, over the total number of cars involved. As one can observe, the Nash equilibrium based solution is never worse than the greedy one. Moreover, by extending the experiment over 100 and 200 executions, our approach is strictly better than the greedy one in the 89% and 93% of the cases respectively, and it allocates the same number of vehicles in the remaining ones.

Since, by construction, a greater number of executions determines a greater number of cars, these experiments also prove the scalability of the algorithm, which seems to behave well with high numbers. Such

a scalability property will be explained in more details in the next section.

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}
	3 slots 3 cars	4 slots 4 cars	5 slots 5 cars	6 slots 6 cars	7 slots 7 cars	8 slots 8 cars	9 slots 9 cars	10 slots 10 cars	11 slots 11 cars	12 slots 12 cars
PSSG	3/3	3/4	5/5	6/6	7/7	8/8	7/9	8/10	9/11	12/12
GPG	2/3	3/4	5/5	5/6	6/7	6/8	6/9	7/10	8/11	10/12

Table 5.2: Resulting vehicle allocations over 10 different simulations applying two solutions to the parking game: the Nash equilibrium based one, and the greedy one.

5.7 Benchmarks

In this section, the benchmark obtained following Algorithm 8 are shown. More precisely, it is analyzed the behavior of the algorithm in the management of a growing number of cars waiting for a parking slot, with respect to a fixed number of parking slots. Two scenarios have been considered:

- In Table 5.3, the number of slots is 4.600. Such a number is not picked at random, but it refers to the number of slots available inside the structure of our case study.

- In Table 5.4, the number of slots is 20.000. Also in this case, the number is not picked at random, but it refers to the number of available slots in the biggest parking space of the world (West Edmonton Mall in Canada).

$slots = 4.000$	
cars	seconds
200	0,001
400	0,002
800	0,004
1.600	0,009
3.200	0,027
6.400	0,402
12.800	1,389
25.600	3,415
51.200	10,165
102.400	33,260
204.800	119,718

Table 5.3: Results on 4.000 slots.

$slots = 20.000$	
cars	seconds
200	0,003
400	0,006
800	0,013
1.600	0,026
3.200	0,060
6.400	0,150
12.800	0,430
25.600	5,687
51.200	23,597
102.400	57,769
204.800	166,093

Table 5.4: Results on 20.000 slots.

Figures 5.3 and 5.4 reflect the quadratic nature of the algorithm: the time is the result of the average between 100 tests. All tests have been executed on an Intel®Core™i5-7300HQ CPU processor of 2,50 GHz, with 8 Gb RAM capacity.

To show the good performance of our algorithm, in our benchmarks we have considered a very large set of cars. The benchmarks show that our tool can be also used in other fields, with much higher numbers. For example, it can be used to accommodate people in a stadium, or, distribute people over hospitals, for example, for a mas-

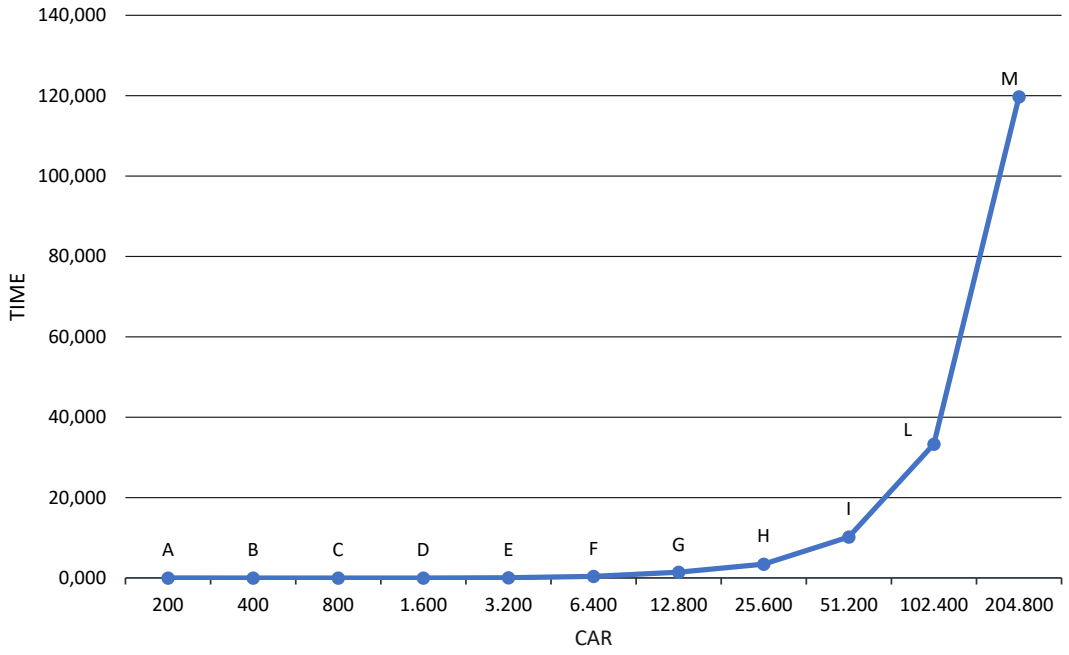


Figure 5.3: Time and cars variation with 4.000 slots.

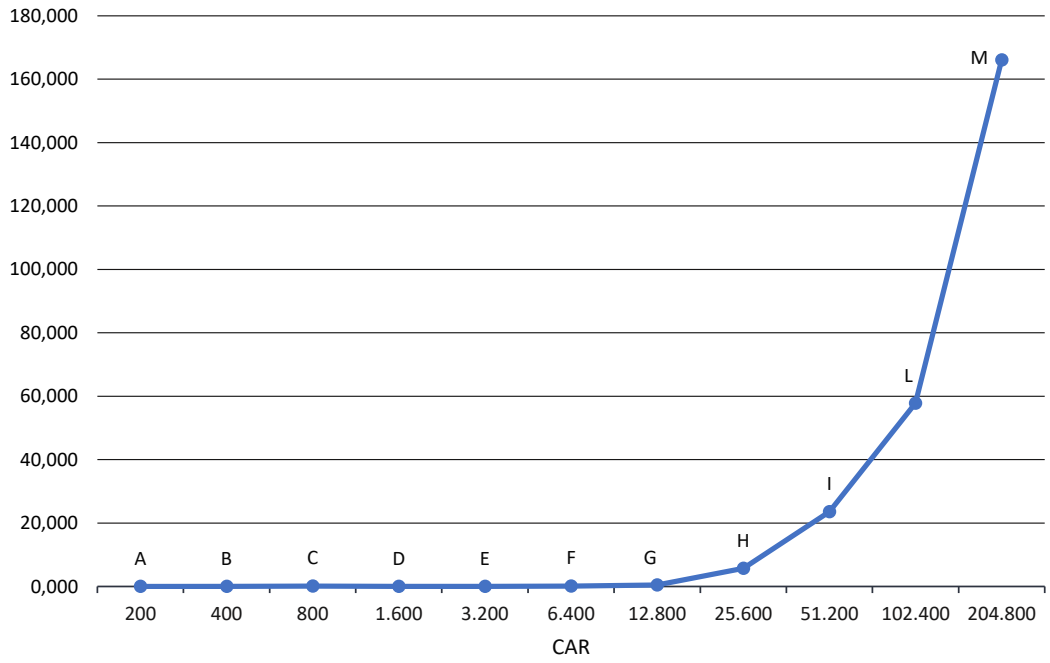


Figure 5.4: Time and cars variation with 20.000 slots.

sive vaccinations, as it is required nowadays for the Covid pandemic situation.

Part V

Conclusions

Chapter 6

Conclusion

Let us move to the conclusions of this work. Vehicular ad hoc networks, and its smart parking application, have been investigated from an algorithmic point of view and a game-theoretic one.

First, a framework for a signal-based representation of a VANET has been provided, as well as a comparison model that produces a similarity value among pairs of signals (networks). Such a representation turns out to be redundancy-free since it just takes vehicles on the x-axis and computes for them a *congestion factor* on the y-axis, and very intuitive: indeed, it is made canonical through a reorganization of the peaks in the signal: each peak is the congestion factor of the corresponding vehicle, and the signal configuration also highlights how the network is divided into clusters. Such a representation lends itself to easy comparison operations: precisely, a similarity function considering the peaks trend of two signals has been defined. This function

has also been compared against a very common algorithm for signal comparison (DTW) and has been proved to be much more precise.

Then, the smart parking problem has been faced in two different flavors: first, the Ant Colony Optimization problem has been mapped to the parking one, by letting drivers looking for an available parking space as ants looking for food. The pheromone, which in ACO setting attracts ants along specific paths, here has been treated as a repulsive, meaning that it pushes drivers not to follow crowded paths. To do so, the graph representing the network has been opportunely modeled and colored. Simulation results prove that the algorithm allocates vehicles in parking slots in a reasonable number of steps.

After, my research has begun more theoretic. In particular, reachability games in DEL game settings have been considered. Some decidability results have been proved, by restricting the set of possible actions for agents to public actions or public announcements. This has been done by considering two different scenarios: first, for *controller syntesis* two entities (the controller and the environment) play in turn while the rest of the agents can only observe the evolution of the game; then, for the *distributed strategy synthesis*, agents are no longer merely observers but can cooperate, against the remaining players, to enforce some objective (expressed as a winning condition in LTLK). The undecidability problem has been solved by assuming public actions and hierarchical information, leading to decidable solutions

also for multi-player DEL games. Moreover, the turn-based scenario has been turned into a concurrent one, by defining a new *concurrent update product* between an epistemic model and an action model. To do so, it has also been necessary to deal with conflicting actions, that have been solved through a *scheduler*.

Speaking about multi-agent systems, it is worthy to do a digression on tools for model checking. One of the most popular is MCMAS, developed by the Imperial College of London. The first version of the tool was thought for ATL and, hence, it also works for model checking of SL formulas. Later, a new version for a fragment of SL, SL[1G], has been developed. The comparative study of this work has focused on a further fragment, namely SL[1G], by trying the two versions of the tool over SL[1G] formulas. Results show that the first version behaves much better than the second one, which suggests probably an ad hoc implementation for the new fragment.

Finally, formal verification methods have been brought into the VANETs world, in particular for smart parking. Precisely, a real scenario has been considered, the Federico II Hospital Company, and a game-based solution for smart parking has been provided. The environment has been modeled as a multi-player game, where drivers are agents that can choose to park in a given slot. The solution has been found as the Nash Equilibrium among players in the game, which turned out to be the best compromise between feasibility and optimal-

ity. A complete simulation for a 3-players-3-slots game has been proposed, while for more complex numbers of players and slots only the time needed has been shown. Moreover, a comparison with a greedy solution has been made, by proving the efficiency of the proposal.

My future research works will follow the lead of what has been done during my PhD, hoping that this approach can be inspiring for researchers to range over several disciplines to make apparently different world meet for stimulating proposals.

Bibliography

[1] CB Abhilash and K Rohitaksha. A comparative study on global and local alignment algorithm methods. *The International Journal of Emerging Technology and Advanced Engineering*, 4(1):1–10, 2014.

[2] Marco Agizza, Walter Balzano, and Silvia Stranieri. An improved ant colony optimization based parking algorithm with graph coloring. In Leonard Barolli, Isaac Woungang, and Tomoya Enokido, editors, *The 36th International Conference on Advanced Information Networking and Applications (AINA-2022) University of Technology Sydney (UTS), Sydney, Australia April 13 - 15, 2022*. Springer, 2022.

[3] Thomas Ågotnes and Hans van Ditmarsch. What will they say? - public announcement games. *Synthese*, 179(Supplement-1):57–85, 2011.

[4] Thomas Ågotnes and Hans van Ditmarsch. What will they say?—public announcement games. *Synthese*, 179(1):57–85,

2011.

[5] Wesam Al Amiri, Mohamed Baza, Karim Banawan, Mohamed Mahmoud, Waleed Alasmary, and Kemal Akkaya. Privacy-preserving smart parking system using blockchain and private information retrieval. In *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pages 1–6. IEEE, 2019.

[6] Othman S Al-Heety, Zahriladha Zakaria, Mahamod Ismail, Mohammed Mudhafar Shakir, Sameer Alani, and Hussein Alsariera. A comprehensive survey: Benefits, services, recent works, challenges, security, and use cases for sdn-vanet. *IEEE Access*, 8:91028–91047, 2020.

[7] Saif Al-Sultan, Moath M Al-Doori, Ali H Al-Bayatti, and Hussien Zedan. A comprehensive survey on vehicular ad hoc network. *Journal of network and computer applications*, 37:380–392, 2014.

[8] Ghulam Ali, Tariq Ali, Muhammad Irfan, Umar Draz, Muhammad Sohail, Adam Glowacz, Maciej Sulowicz, Ryszard Mielnik, Zaid Bin Faheem, and Claudia Martis. Iot based smart parking system using deep long short memory network. *Electronics*, 9(10):1696, 2020.

[9] Ayoub Alsarhan, Abdel-Rahman Al-Ghuwairi, Islam T Almalkawi, Mohammad Alauthman, and Ahmed Al-Dubai. Machine learning-driven optimization for intrusion detection in smart vehicular networks. *Wireless Personal Communications*, 117(4):3129–3152, 2021.

[10] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *JACM*, 49(5):672–713, 2002.

[11] Rajeev Alur, Thomas A Henzinger, and Orna Kupferman. Alternating-time temporal logic. *JACM*, 49(5):672–713, 2002.

[12] K.R. Apt and E. Grädel. *Lectures in game theory for computer scientists*. Cambridge UP, 2011.

[13] Luis Claudio Arcos. The blockchain technology on the music industry. *Brazilian Journal of Operations & Production Management*, 15(3):439–443, 2018.

[14] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[15] Guillaume Aucher, Bastien Maubert, and Sophie Pinchinat. Automata techniques for epistemic protocol synthesis. *arXiv preprint arXiv:1404.0844*, 2014.

[16] Jean-Francois Baffier, Man-Kwun Chiu, Yago Diez, Matias Korman, Valia Mitsou, André Van Renssen, Marcel Roeloffzen, and Yushi Uno. Hanabi is np-hard, even for cheaters who look at their cards. *arXiv preprint arXiv:1603.01911*, 2016.

[17] K. Bakliwal et al. A multi agent system architecture to implement collaborative learning for social industrial assets. *IFAC*, 51(11):1237–1242, 2018.

[18] A. Baltag, L. S Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In *TARK'98*, pages 43–56, 1998.

[19] Marco Balzano, Walter Balzano, Loredana Sorrentino, and Silvia Stranieri. Smart destination-based parking for the optimization of waiting time. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 1019–1027. Springer, 2020.

[20] Walter Balzano, Marco Lapegna, Silvia Stranieri, and Fabio Vitale. Competitive-blockchain-based parking system with fairness constraints. *Soft Computing, DOI number: 10.1007/S00500-022-06888-1*, 2022.

[21] Walter Balzano, Aniello Murano, Loredana Sorrentino, and Silvia Stranieri. Network signal comparison through waves parameters: a local-alignment-based approach. In *2019 IEEE Inter-*

national Symposium on Measurements & Networking (M&N), pages 1–6. IEEE, 2019.

[22] Walter Balzano, Aniello Murano, Loredana Sorrentino, and Silvia Stranieri. A smart compact traffic network vision based on wave representation. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 870–879. Springer, 2019.

[23] Walter Balzano, Aniello Murano, Loredana Sorrentino, and Silvia Stranieri. Behavioral clustering: A new approach for traffic congestion evaluation. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 1418–1427. Springer, 2020.

[24] Walter Balzano and Silvia Stranieri. Lodgp: a framework for support traffic information systems based on logic paradigm. In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 700–708. Springer, 2017.

[25] Walter Balzano and Silvia Stranieri. Acop: an algorithm based on ant colony optimization for parking slot detection. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 833–840. Springer, 2019.

[26] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of multi-agent systems with imperfect information and public actions. In *AAMAS'17*, 2017.

[27] Francesco Belardinelli, Wojciech Jamroga, Damian Kurpiewski, Vadim Malvone, and Aniello Murano. Strategy logic with simple goals: Tractable reasoning about strategies. In *28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 88–94, 2019.

[28] Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. Verification of multi-agent systems with imperfect information and public actions. In *AAMAS*, volume 17, pages 1268–1276, 2017.

[29] Sofia Belkhala, Siham Benhadou, Khalid Boukhdir, and Hicham Medromi. Smart parking architecture based on multi agent system. *Int. J. Adv. Comput. Sci. Appl*, 10:378–382, 2019.

[30] Mario Roberto Folhadela Benevides and Isaque Macalam Saab Lima. Dynamic epistemic logic with communication actions. *Electronic Notes in Theoretical Computer Science*, 344:67–82, 2019.

[31] Raphaël Berthon, Bastien Maubert, and Aniello Murano. Decidability results for atl* with imperfect information and perfect

recall. In Kate Larson, Michael Winikoff, Sanmay Das, and Edmund H. Durfee, editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 1250–1258. ACM, 2017.

[32] Dietmar Berwanger, Anup Basil Mathew, and Marie van den Bogaard. Hierarchical information and the synthesis of distributed strategies. *Acta Informatica*, 55(8):669–701, 2018.

[33] Achim Blumensath and Erich Gradel. Automatic structures. In *Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No. 99CB36332)*, pages 51–62. IEEE, 2000.

[34] Umesh Bodkhe, Sudeep Tanwar, Karan Parekh, Pimal Khanpara, Sudhanshu Tyagi, Neeraj Kumar, and Mamoun Alazab. Blockchain for industry 4.0: A comprehensive review. *IEEE Access*, 8:79764–79800, 2020.

[35] Thomas Bolander and Mikkel Birkegaard Andersen. Epistemic planning for single-and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1):9–34, 2011.

[36] Thomas Bolander, Martin Holm Jensen, and François Schwarzentruber. Complexity results in epistemic planning. In

Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[37] Mohamed Salah Bouassida and Mohamed Shawky. On the congestion control within vanet. In *2008 1st IFIP Wireless Days*, pages 1–5. IEEE, 2008.

[38] Patricia Bouyer. Games on graphs with a public signal monitoring. In *International Conference on Foundations of Software Science and Computation Structures*, pages 530–547. Springer, 2018.

[39] L. Bozzelli, B. Maubert, and S. Pinchinat. Uniform strategies, rational relations and jumping automata. *Inf. Comput.*, 242:80–107, 2015.

[40] Nils Bulling and Wojciech Jamroga. Comparing variants of strategic ability: how uncertainty and memory influence general properties of games. *Autonomous agents and multi-agent systems*, 28(3):474–518, 2014.

[41] Giuseppe Calise, Aniello Murano, and Silvia Stranieri. The parking problem: A game-theoretic solution. *CoRR*, abs/2204.01395, 2022.

[42] Petr Čermák, Alessio Lomuscio, and Aniello Murano. Verifying and synthesising multi-agent systems against one-goal strategy

logic specifications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[43] A.K. Chandra and L.J. Stockmeyer. Alternation. In *FOCS'76*, 1976.

[44] T. Charrier. *Theoretical complexity of reasoning in dynamic epistemic logic and symbolic approach*. PhD thesis, Université de Rennes 1, 2018.

[45] M. J. Coulombe and J. Lynch. Cooperating in video games? impossible! undecidability of team multiplayer games. In *FUN'18*, pages 14:1–14:16, 2018.

[46] Michael J Coulombe and Jayson Lynch. Cooperating in video games? impossible! undecidability of team multiplayer games. *Theoretical Computer Science*, 839:30–40, 2020.

[47] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, Vignesh Kalyanaraman, et al. Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2(6-10):71, 2016.

[48] Luca de Alfaro and Thomas A Henzinger. Concurrent omega-regular games. In *LICS'00*, pages 141–154. IEEE, 2000.

[49] Tiago De Lima. Alternating-time temporal dynamic epistemic logic. *Journal of Logic and Computation*, 24(6):1145–1178, 2014.

[50] Cédric Dégremont, Benedikt Löwe, and Andreas Witzel. The synchronicity of dynamic epistemic logic. In *TARK '11*, 2011.

[51] Marco Dorigo, Mauro Birattari, Christian Blum, Maurice Clerc, Thomas Stützle, and Alan Winfield. *Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings*, volume 5217. Springer, 2008.

[52] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2-3):243–278, 2005.

[53] Gaëtan Douéneau-Tabot, Sophie Pinchinat, and François Schwarzentruber. Chain-monadic second order logic over regular automatic trees and epistemic planning synthesis. *Advances in Modal Logic*, 7, 2018.

[54] Sean R Eddy. What is dynamic programming? *Nature biotechnology*, 22(7):909–910, 2004.

[55] Rik Eshuis and Roel J. Wieringa. A real-time execution semantics for UML activity diagrams. In *FASE'01*, pages 76–90, 2001.

[56] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[57] Bernd Finkbeiner and Sven Schewe. Uniform distributed synthesis. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 321–330. IEEE, 2005.

[58] Maria Fox and Derek Long. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.*, 20:61–124, 2003.

[59] Hector Geffner and Blai Bonet. A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(1):1–141, 2013.

[60] Taher M Ghazal, Mohammad Kamrul Hasan, Muhammad Turki Alshurideh, Haitham M Alzoubi, Munir Ahmad, Syed Shehryar Akbar, Barween Al Kurdi, and Iman A Akour. Iot for smart cities: Machine learning approaches in smart healthcare—a review. *Future Internet*, 13(8):218, 2021.

[61] Samuel Greengard. *The internet of things*. MIT press, 2021.

[62] Mustafa Maad Hamdi, Lukman Audah, Sami Abduljabbar Rashid, Alaa Hamid Mohammed, Sameer Alani, and Ahmed Shamil Mustafa. A review of applications, characteristics and challenges in vehicular ad hoc networks (vanets). In *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–7. IEEE, 2020.

[63] David Harel and Amir Pnueli. On the development of reactive systems. In *Logics and models of concurrent systems*, pages 477–498. Springer, 1985.

[64] Hannes Hartenstein and Kenneth Laberteaux. *VANET: vehicular applications and inter-networking technologies*, volume 1. John Wiley & Sons, 2009.

[65] Paul G Höglund. Parking, energy consumption and air pollution. *Science of the Total Environment*, 334:39–45, 2004.

[66] Jiaxi Hu, Debiao He, Qinglan Zhao, and Kim-Kwang Raymond Choo. Parking management: A blockchain-based privacy-preserving system. *IEEE Consumer Electron. Mag.*, 8(4):45–49, 2019.

[67] Noor Khalled Jallad and Yousef-Awwad Daraghmi. Graph representation for vanets based on hashmap of arraylist and pairs. In *2020 International Conference on Electrical Engineering and Informatics (ICELTICs)*, pages 1–5. IEEE, 2020.

[68] Wojciech Jamroga, Michał Knapik, Damian Kurpiewski, and Łukasz Mikulski. Approximate verification of strategic abilities under imperfect information. *Artificial Intelligence*, 277:103172, 2019.

[69] B. Jioudi et al. e-parking: Multi-agent smart parking platform for dynamic pricing and reservation sharing service. *Journal of Advanced Computer Science and Applications*, 10(11), 2019.

[70] Evangelia Kokolaki, Merkourios Karaliopoulos, and Ioannis Stavrakakis. On the efficiency of information-assisted search for parking space: A game-theoretic approach. In *IWSOS*, 2013.

[71] Tsung-Ting Kuo, Hyeon-Eui Kim, and Lucila Ohno-Machado. Blockchain distributed ledger technologies for biomedical and health care applications. *Journal of the American Medical Informatics Association*, 24(6):1211–1220, 2017.

[72] Marco Lapegna and Silvia Stranieri. Dclu: A direction-based clustering algorithm for vanets management. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 253–262. Springer, 2021.

[73] Sébastien Lê Cong, Sophie Pinchinat, and François Schwarzentruber. Small undecidable problems in epistemic planning. In *IJCAI*, pages 4780–4786, 2018.

[74] Benjamin Leiding, Parisa Memarmoshrefi, and Dieter Hogrefe. Self-managed and blockchain-based vehicular ad-hoc networks. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 137–140, 2016.

[75] T. De Lima. Alternating-time temporal dynamic epistemic logic. *J. Log. Comput.*, 24(6):1145–1178, 2014.

[76] Trista Lin, Hervé Rivano, and Frédéric Le Mouël. A survey of smart parking solutions. *IEEE Transactions on ITS*, 18(12):3229–3253, 2017.

[77] Trista Lin, Hervé Rivano, and Frédéric Le Mouël. A survey of smart parking solutions. *IEEE Transactions on Intelligent Transportation Systems*, 18(12):3229–3253, 2017.

[78] M.L. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabilistic planning. *J. Artif. Intell. Res.*, 9:1–36, 1998.

[79] Beth Logan and Ariel Salomon. A music similarity function based on signal analysis. In *ICME*, pages 22–25, 2001.

[80] Imperial College London. Mcmas.

[81] Xiao-Shan Lu, Ren-Yong Guo, Hai-Jun Huang, Xiaoming Xu, and Jiajia Chen. Equilibrium analysis of parking for integrated daily commuting. *Research in Transportation Economics*, 2021.

[82] Zhaojun Lu, Wenchao Liu, Qian Wang, Gang Qu, and Zhenglin Liu. A privacy-preserving trust model based on blockchain for vanets. *IEEE Access*, 6:45655–45664, 2018.

[83] Krzysztof Małecki. A computer simulation of traffic flow with on-street parking and drivers' behaviour based on cellular automata and a multi-agent system. *Journal of computational science*, 28:32–42, 2018.

[84] Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[85] Vittorio Maniezzo, Luca Maria Gambardella, and Fabio De Luigi. Ant colony optimization (2004). *Cited in New Optimization Techniques in Engineering. Studies in Fuzziness and Soft Computing, vol 141. Springer,Berlin, Heidelberg*, page 62, 2010.

[86] Alexander Matros. Lloyd shapley and chess with imperfect information. *Games and Economic Behavior*, 108:600–613, 2018.

[87] B. Maubert. *Logical foundations of games with imperfect information : uniform strategies*. PhD thesis, University of Rennes 1, France, 2014.

[88] B. Maubert and A. Murano. Reasoning about knowledge and strategies under hierarchical information. In *KR'18*, pages 530–540, 2018.

[89] Bastien Maubert. *Logical foundations of games with imperfect information: uniform strategies*. PhD thesis, Université Rennes 1, 2014.

[90] Bastien Maubert, Aniello Murano, Sophie Pinchinat, François Schwarzentruber, and Silvia Stranieri. Dynamic epistemic logic games with epistemic temporal goals. *arXiv preprint arXiv:2001.07141*, 2020.

[91] Bastien Maubert, Sophie Pinchinat, and François Schwarzentruber. Reachability games in dynamic epistemic logic. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 499–505. ijcai.org, 2019.

[92] J Michael, ALAN Cohn, and Jared R Butcher. Blockchain technology. *The Journal*, 1(7), 2018.

[93] F. Mogavero, A. Murano, and M.Y. Vardi. Reasoning about strategies. In *FSTTCS 10*, LIPIcs 8, pages 133–144. Leibniz, 2010.

[94] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic (TOCL)*, 15(4):1–47, 2014.

[95] Ahmad Mostafa. Vanet blockchain: A general framework for detecting malicious vehicles. *J. Commun.*, 14(5):356–362, 2019.

[96] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[97] Vincent C Müller and Nick Bostrom. Future progress in artificial intelligence: A survey of expert opinion. In *Fundamental issues of artificial intelligence*, pages 555–572. Springer, 2016.

[98] Giang-Truong Nguyen and Kyungbaek Kim. A survey about consensus algorithms used in blockchain. *Journal of Information processing systems*, 14(1):101–128, 2018.

[99] Ayano Okoso, Keisuke Otaki, and Tomoki Nishi. Multi-agent path finding with priority for cooperative automated valet parking. In *ITSC*, pages 2135–2140, 2019.

[100] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[101] Pavel Paclík, Jana Novovicová, and Robert PW Duin. Building road-sign classifiers using a trainable similarity measure. *IEEE Transactions on Intelligent Transportation Systems*, 7(3):309–321, 2006.

[102] Gautam Pal, Gangmin Li, and Katie Atkinson. Multi-agent big-data lambda architecture model for e-commerce analytics. *Data*, 3(4):58, 2018.

[103] A. T. Parameswaran, M. I. Husain, S. Upadhyaya, et al. Is rssi a reliable parameter in sensor localization algorithms: An experimental study. In *In Proc. of F2DA 2009*. IEEE, 2009.

[104] María Pereda et al. Competing for congestible goods: experimental evidence on parking choice. *Scientific reports*, 10(1):1–10, 2020.

[105] G. Peterson, J. Reif, and S. Azhar. Decision algorithms for multiplayer noncooperative games of incomplete information. *Computers & Mathematics with Applications*, 43(1):179–206, 2002.

[106] Gary Peterson, John Reif, and Salman Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers & Mathematics with Applications*, 41(7-8):957–992, 2001.

[107] Gary L Peterson and John H Reif. Multiple-person alternation. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 348–363. IEEE, 1979.

[108] G.L. Peterson and J.H. Reif. Multiple-person alternation. In *FOCS'79*, 1979.

[109] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *FOCS'90*, pages 746–757, 1990.

[110] C Arden Pope Iii, Richard T Burnett, Michael J Thun, Eugenia E Calle, Daniel Krewski, Kazuhiko Ito, and George D Thurston. Lung cancer, cardiopulmonary mortality, and long-term exposure to fine particulate air pollution. *Jama*, 287(9):1132–1141, 2002.

[111] Bernd Puchala. Asynchronous omega-regular games with partial information. In *MFCS'10*, pages 592–603, 2010.

[112] Ramaswamy Ramanujam and Sunil Simon. A communication based model for games of imperfect information. In *International Conference on Concurrency Theory*, pages 509–523. Springer, 2010.

[113] J. Rintanen. Complexity of planning with partial observability. In *ICAPS'04*, pages 345–354, 2004.

[114] Andrzej Ruta, Yongmin Li, and Xiaohui Liu. Robust class similarity measure for traffic sign recognition. *IEEE Transactions on Intelligent Transportation Systems*, 11(4):846–855, 2010.

[115] Sandeep Saharan, Neeraj Kumar, and Seema Bawa. An efficient smart parking pricing system for smart city environment:

A machine-learning based approach. *Future Generation Computer Systems*, 106:622–640, 2020.

[116] Aamir Shahzad, Jae-young Choi, Naixue Xiong, Young-Gab Kim, and Malrey Lee. Centralized connectivity for multiwireless edge computing and cellular platform: A smart vehicle parking system. *Wireless Communications and Mobile Computing*, 2018, 2018.

[117] Rakesh Shrestha, Rojeena Bajracharya, Anish P Shrestha, and Seung Yeob Nam. A new type of blockchain for secure message exchange in vanet. *Digital communications and networks*, 6(2):177–186, 2020.

[118] Pranav Kumar Singh, Roshan Singh, Sunit Kumar Nandi, and Sukumar Nandi. Smart contract based decentralized parking management in its. In *International Conference on Innovations for Community Services*, pages 66–77. Springer, 2019.

[119] M. Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

[120] Xiaoyu Song, Ruopeng Yang, Changsheng Yin, and Bo Tang. A cooperative aerial interception model based on multi-agent system for uavs. In *IAEAC*, volume 5, pages 873–882, 2021.

[121] Statista. Cities with the longest traffic jam delays in europe in 2019, based on average number of hours lost per year, 2019.

[122] Statista. Proportion of population in cities worldwide from 1985 to 2050, 2021.

[123] Larry J. Stockmeyer and Ashok K. Chandra. Provably difficult combinatorial games. *SIAM J. Comput.*, 8(2):151–174, 1979.

[124] TomTom. Traffic index, 2020.

[125] Sabih ur Rehman, M Arif Khan, Tanveer A Zia, and Lihong Zheng. Vehicular ad-hoc networks (vanets)-an overview and challenges. *Journal of Wireless Networking and Communications*, 3(3):29–38, 2013.

[126] Eric Van Damme. *Stability and perfection of Nash equilibria*, volume 339. Springer, 1991.

[127] Wiebe Van der Hoek and Michael Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia logica*, 75(1):125–157, 2003.

[128] Ron van der Meyden. Common knowledge and update in finite environments. *Inf. Comput.*, 140(2):115–157, 1998.

[129] Hans Van Ditmarsch, Wiebe van Der Hoek, and Barteld Kooi. *Dynamic epistemic logic*, volume 337. Springer Science & Business Media, 2007.

[130] Jan Van Eijck, Floor Sietsma, and Yanjing Wang. Composing models. *Journal of Applied Non-Classical Logics*, 21(3-4):397–425, 2011.

[131] Jan van Eijck, Floor Sietsma, and Yanjing Wang. Composing models. *Journal of Applied Non-Classical Logics*, 21(3-4):397–425, 2011.

[132] Nguyen Hoang Viet, Ngo Anh Vien, SeungGwan Lee, and Tae-Choong Chung. Obstacle avoidance path planning for mobile robot based on multi colony ant algorithm. In *First International Conference on Advances in Computer-Human Interaction*, pages 285–289. IEEE, 2008.

[133] Xianwei Wang, Hao Shi, and Chao Zhang. Path planning for intelligent parking system based on improved ant colony optimization. *IEEE Access*, 8:65267–65273, 2020.

[134] Wikipedia. Received signal strength indication, 2021.

[135] Yair Wiseman. Autonomous vehicles. In *Research Anthology on Cross-Disciplinary Designs and Applications of Automation*, pages 878–889. IGI Global, 2022.

[136] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[137] Lixia Xie, Ying Ding, Hongyu Yang, and Xinmu Wang. Blockchain-based secure and trustworthy internet of things in sdn-enabled 5g-vanets. *IEEE Access*, 7:56656–56666, 2019.

[138] Yao-Tsung Yang, Li-Der Chou, Chia-Wei Tseng, Fan-Hsun Tseng, and Chien-Chang Liu. Blockchain-based traffic event validation and trust verification for vanets. *IEEE Access*, 7:30868–30877, 2019.

[139] Quan Yu, Ximing Wen, and Yongmei Liu. Multi-agent epistemic explanatory diagnosis via reasoning about actions. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[140] Sherali Zeadally, Ray Hunt, Yuh-Shyan Chen, Angela Irwin, and Aamir Hassan. Vehicular ad hoc networks (vanets): status, results, and challenges. *Telecommunication Systems*, 50(4):217–241, 2012.

[141] Zinon Zinonos, Panayiotis Christodoulou, Andreas S. Andreou, and Savvas A. Chatzichristofis. Parkchain: An iot parking service based on blockchain. In *15th International Conference on Distributed Computing in Sensor Systems, DCOSS 2019, Santorini, Greece, May 29-31, 2019*, pages 687–693. IEEE, 2019.

[142] Meftah Zouai, Okba Kazar, Belgacem Haba, and Hamza Saouli. Smart house simulation based multi-agent system and internet of things. In *ICMIT*, pages 201–203, 2017.

Bibliography

[1] CB Abhilash and K Rohitaksha. A comparative study on global and local alignment algorithm methods. *The International Journal of Emerging Technology and Advanced Engineering*, 4(1):1–10, 2014.

[2] Marco Agizza, Walter Balzano, and Silvia Stranieri. An improved ant colony optimization based parking algorithm with graph coloring. In Leonard Barolli, Isaac Woungang, and Tomoya Enokido, editors, *The 36th International Conference on Advanced Information Networking and Applications (AINA-2022) University of Technology Sydney (UTS), Sydney, Australia April 13 - 15, 2022*. Springer, 2022.

[3] Thomas Ågotnes and Hans van Ditmarsch. What will they say? - public announcement games. *Synthese*, 179(Supplement-1):57–85, 2011.

[4] Thomas Ågotnes and Hans van Ditmarsch. What will they say?—public announcement games. *Synthese*, 179(1):57–85,

2011.

[5] Wesam Al Amiri, Mohamed Baza, Karim Banawan, Mohamed Mahmoud, Waleed Alasmary, and Kemal Akkaya. Privacy-preserving smart parking system using blockchain and private information retrieval. In *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pages 1–6. IEEE, 2019.

[6] Othman S Al-Heety, Zahriladha Zakaria, Mahamod Ismail, Mohammed Mudhafar Shakir, Sameer Alani, and Hussein Alsariera. A comprehensive survey: Benefits, services, recent works, challenges, security, and use cases for sdn-vanet. *IEEE Access*, 8:91028–91047, 2020.

[7] Saif Al-Sultan, Moath M Al-Doori, Ali H Al-Bayatti, and Hussien Zedan. A comprehensive survey on vehicular ad hoc network. *Journal of network and computer applications*, 37:380–392, 2014.

[8] Ghulam Ali, Tariq Ali, Muhammad Irfan, Umar Draz, Muhammad Sohail, Adam Glowacz, Maciej Sulowicz, Ryszard Mielnik, Zaid Bin Faheem, and Claudia Martis. Iot based smart parking system using deep long short memory network. *Electronics*, 9(10):1696, 2020.

[9] Ayoub Alsarhan, Abdel-Rahman Al-Ghuwairi, Islam T Almalkawi, Mohammad Alauthman, and Ahmed Al-Dubai. Machine learning-driven optimization for intrusion detection in smart vehicular networks. *Wireless Personal Communications*, 117(4):3129–3152, 2021.

[10] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *JACM*, 49(5):672–713, 2002.

[11] Rajeev Alur, Thomas A Henzinger, and Orna Kupferman. Alternating-time temporal logic. *JACM*, 49(5):672–713, 2002.

[12] K.R. Apt and E. Grädel. *Lectures in game theory for computer scientists*. Cambridge UP, 2011.

[13] Luis Claudio Arcos. The blockchain technology on the music industry. *Brazilian Journal of Operations & Production Management*, 15(3):439–443, 2018.

[14] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[15] Guillaume Aucher, Bastien Maubert, and Sophie Pinchinat. Automata techniques for epistemic protocol synthesis. *arXiv preprint arXiv:1404.0844*, 2014.

[16] Jean-Francois Baffier, Man-Kwun Chiu, Yago Diez, Matias Korman, Valia Mitsou, André Van Renssen, Marcel Roeloffzen, and Yushi Uno. Hanabi is np-hard, even for cheaters who look at their cards. *arXiv preprint arXiv:1603.01911*, 2016.

[17] K. Bakliwal et al. A multi agent system architecture to implement collaborative learning for social industrial assets. *IFAC*, 51(11):1237–1242, 2018.

[18] A. Baltag, L. S Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In *TARK'98*, pages 43–56, 1998.

[19] Marco Balzano, Walter Balzano, Loredana Sorrentino, and Silvia Stranieri. Smart destination-based parking for the optimization of waiting time. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 1019–1027. Springer, 2020.

[20] Walter Balzano, Marco Lapegna, Silvia Stranieri, and Fabio Vitale. Competitive-blockchain-based parking system with fairness constraints. *Soft Computing, DOI number: 10.1007/S00500-022-06888-1*, 2022.

[21] Walter Balzano, Aniello Murano, Loredana Sorrentino, and Silvia Stranieri. Network signal comparison through waves parameters: a local-alignment-based approach. In *2019 IEEE Inter-*

national Symposium on Measurements & Networking (M&N), pages 1–6. IEEE, 2019.

[22] Walter Balzano, Aniello Murano, Loredana Sorrentino, and Silvia Stranieri. A smart compact traffic network vision based on wave representation. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 870–879. Springer, 2019.

[23] Walter Balzano, Aniello Murano, Loredana Sorrentino, and Silvia Stranieri. Behavioral clustering: A new approach for traffic congestion evaluation. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 1418–1427. Springer, 2020.

[24] Walter Balzano and Silvia Stranieri. Lodgp: a framework for support traffic information systems based on logic paradigm. In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 700–708. Springer, 2017.

[25] Walter Balzano and Silvia Stranieri. Acop: an algorithm based on ant colony optimization for parking slot detection. In *Workshops of the International Conference on Advanced Information Networking and Applications*, pages 833–840. Springer, 2019.

[26] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of multi-agent systems with imperfect information and public actions. In *AAMAS'17*, 2017.

[27] Francesco Belardinelli, Wojciech Jamroga, Damian Kurpiewski, Vadim Malvone, and Aniello Murano. Strategy logic with simple goals: Tractable reasoning about strategies. In *28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 88–94, 2019.

[28] Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. Verification of multi-agent systems with imperfect information and public actions. In *AAMAS*, volume 17, pages 1268–1276, 2017.

[29] Sofia Belkhala, Siham Benhadou, Khalid Boukhdir, and Hicham Medromi. Smart parking architecture based on multi agent system. *Int. J. Adv. Comput. Sci. Appl*, 10:378–382, 2019.

[30] Mario Roberto Folhadela Benevides and Isaque Macalam Saab Lima. Dynamic epistemic logic with communication actions. *Electronic Notes in Theoretical Computer Science*, 344:67–82, 2019.

[31] Raphaël Berthon, Bastien Maubert, and Aniello Murano. Decidability results for atl* with imperfect information and perfect

recall. In Kate Larson, Michael Winikoff, Sanmay Das, and Edmund H. Durfee, editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 1250–1258. ACM, 2017.

[32] Dietmar Berwanger, Anup Basil Mathew, and Marie van den Bogaard. Hierarchical information and the synthesis of distributed strategies. *Acta Informatica*, 55(8):669–701, 2018.

[33] Achim Blumensath and Erich Gradel. Automatic structures. In *Proceedings Fifteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No. 99CB36332)*, pages 51–62. IEEE, 2000.

[34] Umesh Bodkhe, Sudeep Tanwar, Karan Parekh, Pimal Khanpara, Sudhanshu Tyagi, Neeraj Kumar, and Mamoun Alazab. Blockchain for industry 4.0: A comprehensive review. *IEEE Access*, 8:79764–79800, 2020.

[35] Thomas Bolander and Mikkel Birkegaard Andersen. Epistemic planning for single-and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1):9–34, 2011.

[36] Thomas Bolander, Martin Holm Jensen, and François Schwarzentruber. Complexity results in epistemic planning. In

Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[37] Mohamed Salah Bouassida and Mohamed Shawky. On the congestion control within vanet. In *2008 1st IFIP Wireless Days*, pages 1–5. IEEE, 2008.

[38] Patricia Bouyer. Games on graphs with a public signal monitoring. In *International Conference on Foundations of Software Science and Computation Structures*, pages 530–547. Springer, 2018.

[39] L. Bozzelli, B. Maubert, and S. Pinchinat. Uniform strategies, rational relations and jumping automata. *Inf. Comput.*, 242:80–107, 2015.

[40] Nils Bulling and Wojciech Jamroga. Comparing variants of strategic ability: how uncertainty and memory influence general properties of games. *Autonomous agents and multi-agent systems*, 28(3):474–518, 2014.

[41] Giuseppe Calise, Aniello Murano, and Silvia Stranieri. The parking problem: A game-theoretic solution. *CoRR*, abs/2204.01395, 2022.

[42] Petr Čermák, Alessio Lomuscio, and Aniello Murano. Verifying and synthesising multi-agent systems against one-goal strategy

logic specifications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[43] A.K. Chandra and L.J. Stockmeyer. Alternation. In *FOCS'76*, 1976.

[44] T. Charrier. *Theoretical complexity of reasoning in dynamic epistemic logic and symbolic approach*. PhD thesis, Université de Rennes 1, 2018.

[45] M. J. Coulombe and J. Lynch. Cooperating in video games? impossible! undecidability of team multiplayer games. In *FUN'18*, pages 14:1–14:16, 2018.

[46] Michael J Coulombe and Jayson Lynch. Cooperating in video games? impossible! undecidability of team multiplayer games. *Theoretical Computer Science*, 839:30–40, 2020.

[47] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, Vignesh Kalyanaraman, et al. Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2(6-10):71, 2016.

[48] Luca de Alfaro and Thomas A Henzinger. Concurrent omega-regular games. In *LICS'00*, pages 141–154. IEEE, 2000.

[49] Tiago De Lima. Alternating-time temporal dynamic epistemic logic. *Journal of Logic and Computation*, 24(6):1145–1178, 2014.

[50] Cédric Dégremont, Benedikt Löwe, and Andreas Witzel. The synchronicity of dynamic epistemic logic. In *TARK '11*, 2011.

[51] Marco Dorigo, Mauro Birattari, Christian Blum, Maurice Clerc, Thomas Stützle, and Alan Winfield. *Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings*, volume 5217. Springer, 2008.

[52] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2-3):243–278, 2005.

[53] Gaëtan Douéneau-Tabot, Sophie Pinchinat, and François Schwarzentruber. Chain-monadic second order logic over regular automatic trees and epistemic planning synthesis. *Advances in Modal Logic*, 7, 2018.

[54] Sean R Eddy. What is dynamic programming? *Nature biotechnology*, 22(7):909–910, 2004.

[55] Rik Eshuis and Roel J. Wieringa. A real-time execution semantics for UML activity diagrams. In *FASE'01*, pages 76–90, 2001.

[56] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[57] Bernd Finkbeiner and Sven Schewe. Uniform distributed synthesis. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 321–330. IEEE, 2005.

[58] Maria Fox and Derek Long. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.*, 20:61–124, 2003.

[59] Hector Geffner and Blai Bonet. A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(1):1–141, 2013.

[60] Taher M Ghazal, Mohammad Kamrul Hasan, Muhammad Turki Alshurideh, Haitham M Alzoubi, Munir Ahmad, Syed Shehryar Akbar, Barween Al Kurdi, and Iman A Akour. Iot for smart cities: Machine learning approaches in smart healthcare—a review. *Future Internet*, 13(8):218, 2021.

[61] Samuel Greengard. *The internet of things*. MIT press, 2021.

[62] Mustafa Maad Hamdi, Lukman Audah, Sami Abduljabbar Rashid, Alaa Hamid Mohammed, Sameer Alani, and Ahmed Shamil Mustafa. A review of applications, characteristics and challenges in vehicular ad hoc networks (vanets). In *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–7. IEEE, 2020.

[63] David Harel and Amir Pnueli. On the development of reactive systems. In *Logics and models of concurrent systems*, pages 477–498. Springer, 1985.

[64] Hannes Hartenstein and Kenneth Laberteaux. *VANET: vehicular applications and inter-networking technologies*, volume 1. John Wiley & Sons, 2009.

[65] Paul G Höglund. Parking, energy consumption and air pollution. *Science of the Total Environment*, 334:39–45, 2004.

[66] Jiaxi Hu, Debiao He, Qinglan Zhao, and Kim-Kwang Raymond Choo. Parking management: A blockchain-based privacy-preserving system. *IEEE Consumer Electron. Mag.*, 8(4):45–49, 2019.

[67] Noor Khalled Jallad and Yousef-Awwad Daraghmi. Graph representation for vanets based on hashmap of arraylist and pairs. In *2020 International Conference on Electrical Engineering and Informatics (ICELTICs)*, pages 1–5. IEEE, 2020.

[68] Wojciech Jamroga, Michał Knapik, Damian Kurpiewski, and Łukasz Mikulski. Approximate verification of strategic abilities under imperfect information. *Artificial Intelligence*, 277:103172, 2019.

[69] B. Jioudi et al. e-parking: Multi-agent smart parking platform for dynamic pricing and reservation sharing service. *Journal of Advanced Computer Science and Applications*, 10(11), 2019.

[70] Evangelia Kokolaki, Merkourios Karaliopoulos, and Ioannis Stavrakakis. On the efficiency of information-assisted search for parking space: A game-theoretic approach. In *IWSOS*, 2013.

[71] Tsung-Ting Kuo, Hyeon-Eui Kim, and Lucila Ohno-Machado. Blockchain distributed ledger technologies for biomedical and health care applications. *Journal of the American Medical Informatics Association*, 24(6):1211–1220, 2017.

[72] Marco Lapegna and Silvia Stranieri. Dclu: A direction-based clustering algorithm for vanets management. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 253–262. Springer, 2021.

[73] Sébastien Lê Cong, Sophie Pinchinat, and François Schwarzentruber. Small undecidable problems in epistemic planning. In *IJCAI*, pages 4780–4786, 2018.

[74] Benjamin Leiding, Parisa Memarmoshrefi, and Dieter Hogrefe. Self-managed and blockchain-based vehicular ad-hoc networks. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 137–140, 2016.

[75] T. De Lima. Alternating-time temporal dynamic epistemic logic. *J. Log. Comput.*, 24(6):1145–1178, 2014.

[76] Trista Lin, Hervé Rivano, and Frédéric Le Mouël. A survey of smart parking solutions. *IEEE Transactions on ITS*, 18(12):3229–3253, 2017.

[77] Trista Lin, Hervé Rivano, and Frédéric Le Mouël. A survey of smart parking solutions. *IEEE Transactions on Intelligent Transportation Systems*, 18(12):3229–3253, 2017.

[78] M.L. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabilistic planning. *J. Artif. Intell. Res.*, 9:1–36, 1998.

[79] Beth Logan and Ariel Salomon. A music similarity function based on signal analysis. In *ICME*, pages 22–25, 2001.

[80] Imperial College London. Mcmas.

[81] Xiao-Shan Lu, Ren-Yong Guo, Hai-Jun Huang, Xiaoming Xu, and Jiajia Chen. Equilibrium analysis of parking for integrated daily commuting. *Research in Transportation Economics*, 2021.

[82] Zhaojun Lu, Wenchao Liu, Qian Wang, Gang Qu, and Zhenglin Liu. A privacy-preserving trust model based on blockchain for vanets. *IEEE Access*, 6:45655–45664, 2018.

[83] Krzysztof Małecki. A computer simulation of traffic flow with on-street parking and drivers' behaviour based on cellular automata and a multi-agent system. *Journal of computational science*, 28:32–42, 2018.

[84] Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[85] Vittorio Maniezzo, Luca Maria Gambardella, and Fabio De Luigi. Ant colony optimization (2004). *Cited in New Optimization Techniques in Engineering. Studies in Fuzziness and Soft Computing, vol 141. Springer,Berlin, Heidelberg*, page 62, 2010.

[86] Alexander Matros. Lloyd shapley and chess with imperfect information. *Games and Economic Behavior*, 108:600–613, 2018.

[87] B. Maubert. *Logical foundations of games with imperfect information : uniform strategies*. PhD thesis, University of Rennes 1, France, 2014.

[88] B. Maubert and A. Murano. Reasoning about knowledge and strategies under hierarchical information. In *KR'18*, pages 530–540, 2018.

[89] Bastien Maubert. *Logical foundations of games with imperfect information: uniform strategies*. PhD thesis, Université Rennes 1, 2014.

[90] Bastien Maubert, Aniello Murano, Sophie Pinchinat, François Schwarzentruber, and Silvia Stranieri. Dynamic epistemic logic games with epistemic temporal goals. *arXiv preprint arXiv:2001.07141*, 2020.

[91] Bastien Maubert, Sophie Pinchinat, and François Schwarzentruber. Reachability games in dynamic epistemic logic. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 499–505. ijcai.org, 2019.

[92] J Michael, ALAN Cohn, and Jared R Butcher. Blockchain technology. *The Journal*, 1(7), 2018.

[93] F. Mogavero, A. Murano, and M.Y. Vardi. Reasoning about strategies. In *FSTTCS 10*, LIPIcs 8, pages 133–144. Leibniz, 2010.

[94] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic (TOCL)*, 15(4):1–47, 2014.

[95] Ahmad Mostafa. Vanet blockchain: A general framework for detecting malicious vehicles. *J. Commun.*, 14(5):356–362, 2019.

[96] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[97] Vincent C Müller and Nick Bostrom. Future progress in artificial intelligence: A survey of expert opinion. In *Fundamental issues of artificial intelligence*, pages 555–572. Springer, 2016.

[98] Giang-Truong Nguyen and Kyungbaek Kim. A survey about consensus algorithms used in blockchain. *Journal of Information processing systems*, 14(1):101–128, 2018.

[99] Ayano Okoso, Keisuke Otaki, and Tomoki Nishi. Multi-agent path finding with priority for cooperative automated valet parking. In *ITSC*, pages 2135–2140, 2019.

[100] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[101] Pavel Paclík, Jana Novovicová, and Robert PW Duin. Building road-sign classifiers using a trainable similarity measure. *IEEE Transactions on Intelligent Transportation Systems*, 7(3):309–321, 2006.

[102] Gautam Pal, Gangmin Li, and Katie Atkinson. Multi-agent big-data lambda architecture model for e-commerce analytics. *Data*, 3(4):58, 2018.

[103] A. T. Parameswaran, M. I. Husain, S. Upadhyaya, et al. Is rssi a reliable parameter in sensor localization algorithms: An experimental study. In *In Proc. of F2DA 2009*. IEEE, 2009.

[104] María Pereda et al. Competing for congestible goods: experimental evidence on parking choice. *Scientific reports*, 10(1):1–10, 2020.

[105] G. Peterson, J. Reif, and S. Azhar. Decision algorithms for multiplayer noncooperative games of incomplete information. *Computers & Mathematics with Applications*, 43(1):179–206, 2002.

[106] Gary Peterson, John Reif, and Salman Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers & Mathematics with Applications*, 41(7-8):957–992, 2001.

[107] Gary L Peterson and John H Reif. Multiple-person alternation. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 348–363. IEEE, 1979.

[108] G.L. Peterson and J.H. Reif. Multiple-person alternation. In *FOCS'79*, 1979.

[109] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *FOCS'90*, pages 746–757, 1990.

[110] C Arden Pope Iii, Richard T Burnett, Michael J Thun, Eugenia E Calle, Daniel Krewski, Kazuhiko Ito, and George D Thurston. Lung cancer, cardiopulmonary mortality, and long-term exposure to fine particulate air pollution. *Jama*, 287(9):1132–1141, 2002.

[111] Bernd Puchala. Asynchronous omega-regular games with partial information. In *MFCS'10*, pages 592–603, 2010.

[112] Ramaswamy Ramanujam and Sunil Simon. A communication based model for games of imperfect information. In *International Conference on Concurrency Theory*, pages 509–523. Springer, 2010.

[113] J. Rintanen. Complexity of planning with partial observability. In *ICAPS'04*, pages 345–354, 2004.

[114] Andrzej Ruta, Yongmin Li, and Xiaohui Liu. Robust class similarity measure for traffic sign recognition. *IEEE Transactions on Intelligent Transportation Systems*, 11(4):846–855, 2010.

[115] Sandeep Saharan, Neeraj Kumar, and Seema Bawa. An efficient smart parking pricing system for smart city environment:

A machine-learning based approach. *Future Generation Computer Systems*, 106:622–640, 2020.

[116] Aamir Shahzad, Jae-young Choi, Naixue Xiong, Young-Gab Kim, and Malrey Lee. Centralized connectivity for multiwireless edge computing and cellular platform: A smart vehicle parking system. *Wireless Communications and Mobile Computing*, 2018, 2018.

[117] Rakesh Shrestha, Rojeena Bajracharya, Anish P Shrestha, and Seung Yeob Nam. A new type of blockchain for secure message exchange in vanet. *Digital communications and networks*, 6(2):177–186, 2020.

[118] Pranav Kumar Singh, Roshan Singh, Sunit Kumar Nandi, and Sukumar Nandi. Smart contract based decentralized parking management in its. In *International Conference on Innovations for Community Services*, pages 66–77. Springer, 2019.

[119] M. Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

[120] Xiaoyu Song, Ruopeng Yang, Changsheng Yin, and Bo Tang. A cooperative aerial interception model based on multi-agent system for uavs. In *IAEAC*, volume 5, pages 873–882, 2021.

[121] Statista. Cities with the longest traffic jam delays in europe in 2019, based on average number of hours lost per year, 2019.

[122] Statista. Proportion of population in cities worldwide from 1985 to 2050, 2021.

[123] Larry J. Stockmeyer and Ashok K. Chandra. Provably difficult combinatorial games. *SIAM J. Comput.*, 8(2):151–174, 1979.

[124] TomTom. Traffic index, 2020.

[125] Sabih ur Rehman, M Arif Khan, Tanveer A Zia, and Lihong Zheng. Vehicular ad-hoc networks (vanets)-an overview and challenges. *Journal of Wireless Networking and Communications*, 3(3):29–38, 2013.

[126] Eric Van Damme. *Stability and perfection of Nash equilibria*, volume 339. Springer, 1991.

[127] Wiebe Van der Hoek and Michael Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia logica*, 75(1):125–157, 2003.

[128] Ron van der Meyden. Common knowledge and update in finite environments. *Inf. Comput.*, 140(2):115–157, 1998.

[129] Hans Van Ditmarsch, Wiebe van Der Hoek, and Barteld Kooi. *Dynamic epistemic logic*, volume 337. Springer Science & Business Media, 2007.

[130] Jan Van Eijck, Floor Sietsma, and Yanjing Wang. Composing models. *Journal of Applied Non-Classical Logics*, 21(3-4):397–425, 2011.

[131] Jan van Eijck, Floor Sietsma, and Yanjing Wang. Composing models. *Journal of Applied Non-Classical Logics*, 21(3-4):397–425, 2011.

[132] Nguyen Hoang Viet, Ngo Anh Vien, SeungGwan Lee, and Tae-Choong Chung. Obstacle avoidance path planning for mobile robot based on multi colony ant algorithm. In *First International Conference on Advances in Computer-Human Interaction*, pages 285–289. IEEE, 2008.

[133] Xianwei Wang, Hao Shi, and Chao Zhang. Path planning for intelligent parking system based on improved ant colony optimization. *IEEE Access*, 8:65267–65273, 2020.

[134] Wikipedia. Received signal strength indication, 2021.

[135] Yair Wiseman. Autonomous vehicles. In *Research Anthology on Cross-Disciplinary Designs and Applications of Automation*, pages 878–889. IGI Global, 2022.

[136] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[137] Lixia Xie, Ying Ding, Hongyu Yang, and Xinmu Wang. Blockchain-based secure and trustworthy internet of things in sdn-enabled 5g-vanets. *IEEE Access*, 7:56656–56666, 2019.

[138] Yao-Tsung Yang, Li-Der Chou, Chia-Wei Tseng, Fan-Hsun Tseng, and Chien-Chang Liu. Blockchain-based traffic event validation and trust verification for vanets. *IEEE Access*, 7:30868–30877, 2019.

[139] Quan Yu, Ximing Wen, and Yongmei Liu. Multi-agent epistemic explanatory diagnosis via reasoning about actions. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[140] Sherali Zeadally, Ray Hunt, Yuh-Shyan Chen, Angela Irwin, and Aamir Hassan. Vehicular ad hoc networks (vanets): status, results, and challenges. *Telecommunication Systems*, 50(4):217–241, 2012.

[141] Zinon Zinonos, Panayiotis Christodoulou, Andreas S. Andreou, and Savvas A. Chatzichristofis. Parkchain: An iot parking service based on blockchain. In *15th International Conference on Distributed Computing in Sensor Systems, DCOSS 2019, Santorini, Greece, May 29-31, 2019*, pages 687–693. IEEE, 2019.

[142] Meftah Zouai, Okba Kazar, Belgacem Haba, and Hamza Saouli. Smart house simulation based multi-agent system and internet of things. In *ICMIT*, pages 201–203, 2017.

Ringraziamenti

Ringrazio Nello e Walter per la fiducia che hanno sempre riposto in me, ma soprattutto per averci creduto anche quando io stessa ho vacillato.

Ringrazio la mia meravigliosa famiglia, i miei genitori, le mie sorelle, le mie nonne, Iaia, per avermi fatta diventare quella che sono oggi, per avermi trasmesso la determinazione e i valori giusti. Siete una parte essenziale della mia vita.

Ringrazio Luca e Martina per esserci stati nei momenti divertenti del mio percorso, per le esperienze (costruttive e non) che abbiamo condiviso e per quelle che ancora ci aspettano.

Ringrazio Erco, Stefano e Bocci per la vostra amicizia sincera e perché so di poter contare su di voi. Sono contenta di avervi nella mia vita.

Ringrazio tutte le lupe, perché la nostra tana é stato il rifugio perfetto per staccare dalle difficoltá. Condividerla con voi é bellissimo.

Ringrazio Lello e Patrizia per esserci sempre e per la loro generositá e ringrazio Mirella, la mia terza sorella, per il legame che si é creato sin da subito e per aver scelto di condividere con me uno dei momenti piú belli della sua vita. Vi voglio bene.

Infine, ringrazio Marco, mio amico, coinquilino, confidente, compagno, fidanzato, qualche volta figlio, ma sempre presente nella mia vita. Grazie per aver partecipato a questo percorso dall'inizio. Con la tua capacitá di strapparmi sempre un sorriso, non sai quante giornate

”no” hai migliorato. Grazie perché la nostra mini famiglia con Emmuz mi da tutti i giorni gli stimoli giusti per lavorare a raggiungere i miei obiettivi. Tutti i traguardi che verranno saranno anche tuoi. Ti amo.