Thése de doctorat

Pour obtenir le grade de Docteur de

l'Université Polytechnique Hauts-de-France

Discispline : **Informatique**

Présentée et soutenue par : MARKO MLADENOVIC.

Le 14/01/2020, à Valenciennes.

École doctorale : Sciences Pour l'Ingénieur (ED SPI 072).
Laboratoire : LAMIH UMR CNRS 8201.

LE PROBLÈME D'AFFECTATION DE PLACES DE STATIONNEMENT DYNAMIQUE : SOLUTIONS PRATIQUES ET THÉORIQUES

Président du jury	:	BRUNO DEFUDE.	Professeur, Institut Mines-Telecom.
Rapporteurs	:	FRANÇOIS CLAUTIAUX.	Professeur, Université de Bordeaux.
		LEANDRO COELHO.	Professeur, Université de Laval.
Examinateurs	:	AGNÈS PLATEAU.	Maître de Conférences, CNAM Paris.
		CAROLINE PRODHON.	Maître de Conférences HDR, UTT.
Directeurs de thèse	:	THIERRY DELOT.	Professeur, UPHF.
		GILBERT LAPORTE.	Professeur, HEC Montréal.
Co-encadrant de thèse	:	CHRISTOPHE WILBAUT.	Maître de Conférences, UPHF.

Janvier 2020

Thése de doctorat

Pour obtenir le grade de Docteur de

l'Université Polytechnique Hauts-de-France

Discispline : **Informatique**

Présentée et soutenue par : MARKO MLADENOVIC.

Le 14/01/2020, à Valenciennes.

École doctorale	:	Sciences Pour l'Ingénieur (ED SPI 072).
Laboratoire	:	LAMIH UMR CNRS 8201.

THE DYNAMIC PARKING ALLOCATION PROBLEM : THEORETICAL AND PRACTICAL SOLUTION METHODS

Président du jury	:	BRUNO DEFUDE.	Professeur, Institut Mines-Telecom.
Rapporteurs	:	FRANÇOIS CLAUTIAUX.	Professeur, Université de Bordeaux.
		LEANDRO COELHO.	Professeur, Université de Laval.
Examinateurs	:	AGNÈS PLATEAU.	Maître de Conférences, CNAM Paris.
		CAROLINE PRODHON.	Maître de Conférences HDR, UTT.
Directeurs de thèse	:	THIERRY DELOT.	Professeur, UPHF.
		GILBERT LAPORTE.	Professeur, HEC Montréal.
Co-encadrant de thèse	:	CHRISTOPHE WILBAUT.	Maître de Conférences, UPHF.

Janvier 2020

RÉSUMÉ

Cette thèse s'intéresse à la problématique du stationnement en milieu urbain, en particulier en période de forte influence. Un des objectifs principaux est l'exploration de méthodes issues de la recherche opérationnelle pour apporter des solutions pratiques via la programmation mathématique et les heuristiques. Nous considérons dans un premier temps une version simplifiée, statique du problème, dans laquelle l'ensemble des données nécessaires est fixé et déterministe sur un horizon d'une journée. Un modèle en variables 0-1, dérivé du problème d'affectation généralisée est proposé et testé sur un ensemble d'instances généré aléatoirement. Des extensions qui incluent des fonctions objectives différentes et d'autres modes de transport sont aussi examinés. De plus, nous avons proposé une heuristique basée sur la recherche à voisinage variable pour obtenir rapidement une solution de qualité satisfaisante. La nature dynamique du problème nous a conduit à adapter ce modèle de façon à pouvoir prendre en compte les mise à jour continues des données. Nous avons ainsi proposé et évalué plusieurs politiques et scénarios, avec l'ambition d'obtenir un système qui soit le plus adaptatif et robuste possible. Le système proposé doit pouvoir guider les utilisateurs vers une place qui leur est affectée lorsque cela est possible, ou à défaut vers leur destination. Notre approche est validée via un ensemble de simulations réalisées à partir de données réelles collectées depuis trois grandes villes européennes.

Mots clés: Parking, Optimisation combinatoire, Affectation, Recherche à voisinage variable, Programmation en variables 0-1, Problèmes dynamiques.

ABSTRACT

This thesis focuses on the problem of urban parking, especially in peak traffic hours. One of the main objectives is to explore the solution methods from operational research perspective and to provide practical solutions through mathematical programming and heuristics. We first consider a simplified, static version of the problem in which all the necessary data is fixed and deterministic over a one-day planning horizon. A 0-1 programming model derived from the generalized assignment problem is proposed and tested on a randomly generated set of instances. Extensions that include different objective functions and other modes of transport are also examined. In addition, we proposed a heuristic based on variable neighborhood search to quickly obtain a good quality solutions. The dynamic nature of the problem has led us to adapt this model so that it can take into account the continuous data updates. We have proposed and evaluated several policies and scenarios, with the goal of developing a system that is as adaptive and robust as possible. The proposed system should be able to guide users to a parking lot assigned to them when possible, or to their destination when their is no parking slot available. Our approach is corroborated via simulation over a set of real data collected from three major European cities.

Keywords: Parking, Combinatorial optimization, Dynamism, Assignment, Variable neighborhood search, 0-1 programming.

CONTENTS

LIST OF ABBREVIATIONS

AP	Assignment Problem
AV	Autonomous Vehicle
B&B	Branch and Bound
DAP	Dynamic Assignment Problem
DGAP	Dynamic Generalized Assignment Problem
DPAP	Dynamic Parking Allocation Problem
DVRP	Dynamic Vehicle Routing Problem
EV	Electric Vehicle
GAP	Generalized Assignment Problem
GPS	Global Positioning System
IoT	Internet of Things
IPS	Indoor Positioning System
LS	Local Search
MIP	Mixed Integer Programming
MOCO	Multi-objective combinatorial optimization
MP	Mathematical Programming
NP	Non polynomial
OR	Operations Research
PAP	Parking Allocation Problem
PDP	Pickup and Delivery Problem
PDPTW	PDP with Time Windows
PGI	Parking Guidance and Information
PPP	Parking Pricing Problem
P+R	Park-and-ride
SOP	Stochastic Optimization Problem
TU	Totally Unimodular
TS	Tabu Search
TSP	Traveling Salesman Problem
TVTSP	Time varying TSP
VANET	Vehicular ad-hoc Network
VRPTW	Vehicle Routing Problem with Time Windows
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem

PREFACE

Cette thèse est réalisée en partenariat entre le LAMIH et le CIRRELT dans le cadre du Laboratoire International Associé (LIA) Recherche Opérationnelle et Informatique en Transport Mobilité et Logistique (LIA-ROI-TML) du CNRS. Le LIA regroupe deux centres de recherche, un en France, le Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines (LAMIH UMR CNRS 8201) et un au Canada, le Centre interuniversitaire de recherche sur les réseaux d'entreprises, la logistique et le transport (CIRRELT). La direction de la thèse a été menée par le Professeur Thierry DELOT du côté du LAMIH (à l'Université Polytechnique Hauts-de-France) et par le Professeur Gilbert LAPORTE du côté CIRRELT, HEC Montréal.

ACKNOWLEDGMENTS

Peu personnes sont capables de réussir tout seuls et je ne suis aucune exception. J'ai partiellement réussi à remercier toutes les personnes qui m'ont aidé au long de ma thèse avec quelques communications scientifiques. Mais ils méritent beaucoup plus que ça. Cette page est dédiée à eux en espérant que je serai à la hauteur de retourner la faveur quand le temps viendra.

Tout d'abord, je voudrais remercier la région Hauts-de-France qui a co-financé cette thèse et sans laquelle celle-ci n'aurait pas été possible. Du côté du laboratoire et de l'Université, je dois particulièrement remercier mes encadrants : professeur Thierry DELOT et dr. Christophe WILBAUT. Je me souviens toujours de notre première réunion et je me pose souvent la question si j'ai pu faire mieux. Pour leurs patience, conseils et soutien, je serai toujours reconnaissant. Une autre personne du laboratoire m'a particulièrement soutenu : professeur Saïd HANAFI qui était d'ailleurs mon premier contact avec le monde scientifique et sur qui je pouvais toujours compter. L'injustice des remerciements se reflète dans le choix des noms qui vont se retrouver dans la liste et je voudrais dire que je remercie tous mes collègues du laboratoire et tous le personnels qui était souvent présent pour moi. De tous ces gens, je me sens obligé de remercier deux personnes en particulier : Maureen COURBEZ pour toute son aide et Ayoub NEGGAZ, un vrai ami.

Si quelqu'un m'aurait dit avant le début de ma thèse que professeur Gilbert LAPORTE serait un de mes directeurs de thèse, je ne l'aurai pas cru. Même à ce jour, j'ai du mal à y croire. Je ne peux pas lui remercier assez. Pour ces conseils, sa manière d'approcher la recherche et son soutien. De plus, prof. LAPORTE partira bientôt en retraite et je serai parmi ses derniers doctorants. Quel privilège.

Mon feu grand-père avait l'habitude de dire que « la famille est la cellule de la société ». Ça m'a pris beaucoup de temps pour vraiment comprendre ses mots. À mon père sans lequel je n'aurais probablement jamais poursuivi une voie scientifique, à ma mère qui est toujours prête à discuter sur des sujets scientifiques et qui pose des questions difficiles, à ma sœur qui voit toujours plus loin et qui reste le moteur créative de notre famille ; merci, merci.

Je dois terminer les remerciements en demandant pardon. Pardon à tous avec qui je n'étais pas toujours poli, avec qui j'étais souvent de mauvaise humeur et pour tous les retards. Certains sont parti et je les comprends. Je n'étais pas toujours à la hauteur et je vous prie de m'excuser.

THESIS OVERVIEW

Parking related problems present a know headache for most drivers and are at the source of traffic congestion in many cities, producing centuries of lost time in total, millions of tons of emissions, and consequently huge economical losses. Moreover, as we are stepping in the era of new types of vehicles, such as the electric vehicles and autonomous vehicles that will reduce emissions, however the complication of where to park them will still remain. In this thesis, we investigate solution methods to this problem in an attempt to improve the way vehicles are assigned to parking lots in urban areas, a problem we call the parking allocation problem. More precisely, we consider the problem of assigning optimal parking lots to a set of vehicles in a city over a given planning horizon. The problem may seem as a simple type of matching or assignment problem, but our review of the literature shows that there is no standard formulation for this problem proposed by the community. Nevertheless there were numerous efforts to formulate and solve this problem, but because of its high level of dynamism and stochasticity, there is still no standardized way to address it. What makes it complex is the fact that whenever we assign parking to a vehicle at a given time, the traffic, number of vehicles, number of available slots and other factors may have completely changed since we received the initial requests. Thus, the previous allocations could be unsuitable for the current vehicle/traffic configuration. This is a typical consequence of dynamic problems and the parking allocation problem fits the bill perfectly. Therefore, we are looking at a familiar issue in operations research, but which has not yet been very well charted, i.e., there is no clear consensus on how to treat dynamic problems in the field of operations research.

The goal of this thesis is to propose a combinatorial optimization formulation for the parking allocation problem, but also make it capable to process and reevaluate new input as fast as possible, i.e., in near real time. The approach we opted for to tackle this topic is threefold:

1. propose a generic combinatorial formulation for the static case,
2. develop a mechanism capable of coping with the dynamic changes,
3. develop a testing environment to evaluate our approach.

In this chapter we briefly introduce these three steps of our approach and present the main contributions and outline of the thesis.

Combinatorial optimization formulation

The ultimate goal of this thesis is to propose a system that could be put in practical use, and at the same time, is theoretically justified. In order to get there, we first need to introduce a combinatorial optimization problem that handles the static case, i.e., to assign parking lots to vehicles at a fixed time moment. This formulation needs to include arrival times of the vehicles to ensure that there will be an available slot at the time of their arrival. This makes the formulation time-dependent from the start, but not yet capable of coping with dynamic input updates. Furthermore, already at the static level we anticipate myopic traps, in the form that we take into account the number of available slots at the arrival time of each vehicle. The assumptions and requirements to state the problem and populate its input are minimalistic and rely mainly on the inter-connectivity of a set of vehicles. This set of vehicles represents only a subset of vehicles in traffic that use our parking allocation system, i.e., we do not assume that information of all the vehicles in traffic is known. Still, our formulation includes the necessary constraints: allocation and capacity. Our goal now is to determine the optimal parking lot for this set of vehicles, respecting these two constraints, while minimizing the total traveling time. The traveling time includes the time needed to reach their parking and then the time required to reach their destination on foot. Since we do not assume control over all the vehicles in traffic, we can not impose flow constraints, i.e., estimate how long a vehicle would remain parked, because we just know the number of parked vehicles at a given time, but not which vehicle entered and which left.

The formulation is based on an assignment problem variant, the generalized assignment problem, which is in general NP-hard. However, by design, we made our 0-1 programming model to possess the integrality property, thus making it relatively easy to solve optimally with commercial solvers. Nevertheless if the vehicle or parking number is very large, the solver requires a significant amount of time to build the model, and therefore heuristics represent a good alternative when a large-scale instance is considered. For this purpose we developed two heuristics for solving the proposed model,

along with a commercial solver.

This model corresponds to the static case of the parking allocation problem (PAP) and represents the first step in the overall process. This means that it attributes parking lots to vehicles for a given input, and cannot handle continuous input updates. However, since we have several efficient solving techniques, exact and approximate, we can go a step further and base our dynamic mechanism around this model. More precisely, the PAP model can serve as a tool for allocating parking lots within a dynamic configuration.

Dynamic mechanism

For a given planning horizon, we wish to coordinate vehicles to parking facilities with respect to the current traffic situation. More accurately, adapt to changes which occur during that planning horizon. The parking allocation is decided at a fixed time step by the static PAP model. In order to integrate changes, we keep all the vehicles in a list of active vehicles until they reach their parking lots. This allows us to reevaluate previous decisions for each vehicle. This cannot be possible if the allocation process takes a lot of time. Furthermore, if there is no available spot for some vehicle, we guide it towards its destination. However, at some future point of the planning horizon, a spot can be made available and we would be able to guide that vehicle towards it. Keeping all the vehicles active until reaching their parking lots has another advantage: it avoids allocating the same spot to two vehicles appearing at different points of the planning horizon. This concept represents the basis of the mechanism that we developed in this thesis.

Applying this idea, we are able to deploy a sequence of static PAPs over the planning horizon. We call this sequence with its active vehicles the dynamic PAP (DPAP). The precise moment when a PAP is deployed within the DPAP sequence and under which rules is characterized by policies which can be added to maintain a strategic goal. These policies do not interfere with the complexity of the PAP and serve as general guidelines for different scenarios. Moreover, they are not function to be determined, but fixed in advance and empirically tested.

All of these steps of the DPAP sequence are reunited in a framework that we call the DPAP framework. This framework is comprised of four layers that correspond to

the overall policies, static PAP, the sequence design, and real data. The real data layer is included to avoid many random variables and produce more accurate input. More precisely, the DPAP framework is based on the online algorithm within the dynamic mechanism of the DPAP framework, solving static PAPS at each decision moment of the overall planning horizon, following general guidelines and restrictions imposed by a fixed policy and being fed by accurate real data.

Testing environment

It would be overly optimistic to consider that data of all the vehicles in traffic are available. This is why we assume just partial knowledge of the overall vehicle set. This makes us rely on the real parking availability data.

Parking availability information can be found from public sources together with its real-time evolution. By collecting parking availability data from three cities, we were able to apply our mechanism and evaluate its performance. The three cities are: Belgrade (Serbia), Luxembourg City (Luxembourg), and Lyon (France).

We were further able to extract geographical coordinates of those parking facilities and the city maps. This allowed us to develop a simulation environment in which vehicles appear. The vehicles are characterized only by their current geographical position and their destination. Based only on their destinations and the real data, we are then able to run the DPAP and offer a parking spot if possible to all the vehicles in near real time.

If there is no parking available for a vehicle at a given point of the planning horizon, we direct it to its destination. By doing so, we are buying time so that a parking slot can be liberated in some future point of the planning horizon. Furthermore, this does not deteriorate the vehicle's route, since it would anyway be headed to its destination.

Contributions

The literature offers a multitude of mathematical programming (MP) models adapted to the problem of allocating parking lots to vehicles. Some have unreasonable or excessive assumptions, others are entirely static. Then, some models are very complex, and require a significant amount of computing power. Therefore the first order of business was to find the most influential papers on the topic. To do so, we start with the simplest

problems, such as the assignment problem (AP) and consider its dynamic analog, and build-up to more elaborate formulations, such as the vehicle routing problem (VRP). An in-depth survey of recent and most significant papers on the topic is classified and analyzed. The first contribution of this thesis is to propose a new classification of the PAP. We draw a parallel to other dynamic problems in transportation to first classify papers using an existing classification of the dynamic VRP, and then introduce a new classification specific to the PAP. We propose a classification of the existing literature concerning the PAP.

The second contribution was to propose an efficient MP model to the PAP with realistic assumptions which would allocate parking lots to vehicles. We discovered that if we include the allocations to be time-dependent then the model could be made to possess the integrality property and thus can be quickly solved with commercial solvers. Even so, if the number of parking or vehicles is very large, it consumes a lot of time. For this purpose, a variable neighborhood search heuristic was developed and used to solve the proposed model. Several properties of the model and heuristics are detailed and tests on sets of up to 90,000 vehicles were conducted.

In this thesis we address the topic of highly dynamic and stochastic problems via the PAP. We opted for a simple, yet effective mechanism to tackle this uncharted terrain of operations research. The third contribution of the thesis is introducing and successfully justifying the DPAP framework that is able to deal with the dynamic changes.

The body of literature regarding dynamic formulations does not provide a way to evaluate solutions, as it does in static formulations. In a static problem the goal is to minimize or maximize the objective function (single or multi objective). However, dynamic solutions cannot be compared in this fashion. This is why we developed an environment, based on real data capable of simulating the DPAP. Furthermore, we propose a way to evaluate the dynamically obtained solutions.

Thesis outline

This manuscript is divided into six chapters. Chapter 1 presents an overview of several aspects of parking related problems, such as parking allocation, pricing, consequences on traffic, etc. It also defines the specific problem that will be studied in this

thesis. Once the problem is defined, Chapter 2 surveys the state of the art and positions our contributions within the literature. It investigates both static and dynamic formulations of the PAP by gradually presenting first simple optimization problems and their dynamic analog, and then more complex problems and their dynamic versions. In order to solve the dynamic PAP, we first need to tackle the static version, that is introduced in Chapter 3. The problem is defined, and a corresponding 0-1 programming model is proposed. Detailed analysis on both the model and the solving techniques is also discussed. Chapter 4 addresses the dynamic case of the PAP. The dynamic PAP is solved by introducing a flexible framework that handles dynamic changes in near real-time. Moreover, the chapter provides several policies and scenarios and evaluates them on real data gathered in three European cities. A significant effort was invested in creating a simulation environment that can evaluate the results obtained by the DPAP framework. Thus Chapter 5 is devoted to the geographical and physical data which were used to develop the simulation, as well as the parameters that were required for it. Furthermore, the main results of this thesis are summarized in this chapter. Chapter 6 closes the thesis with the concluding remarks and future work perspectives.

CHAPTER 1

PARKING RELATED PROBLEMS: A GENERAL CONTEXT

The effects of a massive unorganized pursuit of parking represents a well-know headache for most drivers. On a large-scale all the adverse effects of a chaotic parking pursuit become much more detrimental. What is less known is that the problem of finding a parking slot dates from the beginning of the 20^{th} century. Historically, parking meters were the first attempt of managing parking. Over time and with technological breakthroughs, adequate applications were put into use for better managing the parking-related problems. Today a wide range of commercial solutions are provided for commuters in search for a place to park. In parallel, the scientific community has studied various effects of parking and investigated different methods that could improve the search for parking, such as reservation systems and variable parking tariffs. However, one may argue that the problem of finding a convenient parking slot has yet to find a true solution. New types of vehicles bring with them new constraints into consideration when looking for an available parking space. For example, upcoming autonomous vehicles and the growing number of electric vehicles also require a space to park. Thus, though alleviating adverse effects and offering more efficient engines and reducing pollution, the question of where to park them and what to do with them while parked, remains present. Moreover, it is a serious topic of discussion in both scientific and industrial circles.

The problem itself can be viewed and thus tackled in various ways: by providing pricing incentives for some parking lots, by making use of vehicle connectivity to guide drivers, by introducing general policies which restrict the number of vehicles in a certain zone, etc. In other words, the perspective on the topic will be influenced by the intention of the decision maker: maximize revenue, minimize traffic congestion, maximize social welfare, etc. Therefore, before defining the specific problem studied in this manuscript, it is useful to gain an insight in the diversity of the parking-related problems. This chapter provides such an insight. First by underlining the magnitude of the negative effects caused by the blind search for parking. Then by addressing some of the techniques applied to mitigate these adverse effects. Therefore, this chapter makes a broad recap of various parking-related problems and provides transportation and urbanistic details

on how far the implications of unorganized parking search may reach. Chapter 2 then surveys in more detail the main topic of interest: parking assignment.

Chapter outline All the previously mentioned aspects surrounding parking management are addressed in this chapter. Section 1.1 provides information on how the pursuit of parking affects the traffic, environment, and economics. It also includes some practical commercial solutions that are available in most urban areas. Historically, the first attempt to organize parking in urban areas was by introducing the parking meter. Therefore, researchers studied how different parking tariffs affect the traffic. Section 1.2 takes a look at the parking pricing by surveying some recent and more influential papers on this subject. In recent years, most papers dealing with the topic of parking management are examined under the umbrella of internet of things, most commonly in the context of smart cities. Section 1.3 of this chapter addresses those papers. Section 1.4 is dedicated to various mathematical programming (MP) formulation of the parking allocation problem. Here we briefly survey the literature in order to pin down where our results fit into the literature and define the main topic of this manuscript: parking allocation. The chapter is concluded in Section 1.5.

1.1 Effects of blind parking pursuit in urban areas

Most drivers found themselves at least once circling around looking for an available parking slot with no success. This is a recurring theme in most big cities. However, the repercussions of this effect, when all vehicles are taken into consideration are immense. First of all, different countries have different traffic (parking) regulations, and deal with this problem following those regulations. However, it represents a major traffic factor in all cases. A less known fact is that parking-related problems stretch back to the period of first mass car production, namely the beginning of the 20^{th} century, predating operational research (OR). An interesting quote which illustrates well this fact is from the American City magazine in 1920: "*The right to move a car is superior to the right to store cars on the public way*", see [103]. Moreover, we see the first serious studies regarding parking also appearing in the 1920s, see Table 1.I. For example, it was estimated that around one third of vehicles in a district of Detroit were in fact just cruising for parking.

In more modern times, one of the authors who drew the most attention to the consequences of unorganized parking was Donald Shoup. In one of his studies Shoup [99], reveals that the search for vacant curb parking, even thought they may be cheaper, does not pay off. This is because other criteria should be taken into consideration, such as the time spent searching (cruising) for parking at the curb, fuel cost of cruising, the number of people in the car, parking duration, etc. He then proposed different pricing techniques and advocated the use of "off-street parking" as a better alternative to a random street search. Off-street parking represents a facility (indoor or outdoor) capable of accommodating vehicles for some time period, namely a *parking lot*, or *car park* in British English, see Figure 1.1. In another paper, [98], Shoup claimed that, cumulatively for one year, in just one district of Los Angeles around 47,000 gallons of gasoline were burned producing 730t of CO_2 and taking drivers 945,000 extra miles (for a total of 11 years) to find a vacant slot. Ayala *et al.* [7] report that each year in Chicago 63 million miles are travelled searching for a vacant space, which in turn generates 48,000 tons of carbon dioxide. Similar observations were reported in [5, 6, 35, 104] highlighting the effects parking cruising has on the level of traffic congestion and environmental quality within an urban center.

Previously mentioned papers based their observations on data mainly collected from

Year	City	Share of traffic cruising	Average search time (in minutes)
1927	Detroit (1)	19%	/
1927	Detroit (2)	34%	/
1933	Washington	/	8.0
1960	New Haven	17%	/
1965	London (1)	/	6.1
1965	London (2)	/	3.5
1965	London (3)	/	3.6
1977	Freiburg	74%	6.0
1984	Jerusalem	/	9.0
1985	Cambridge	30%	11.5
1993	Cape Town	/	12.2
1993	New York (1)	8%	7.9
1993	New York (2)	/	10.2
1993	New York (3)	/	13.9
1997	San Francisco	/	6.5
2001	Sydney	/	6.5
	Average	30%	8.1

Table 1.I – Effects of cruising for parking in the 20^{th} century. The numbers in parentheses after Detroit, London, and New York refer to different locations within the same city [97].

the USA where curbside parking is more common than in other countries. For example, on-street parking represents 63% and 75.5% [1] of Los Angeles' and San Francisco's entire parking capacities, respectively. A more moderate percentage is reported in European cities where 37% [2] of parking capacities are on-street and only 5% in Beijing [3]. A study by Gantelet & Lefauconnier [37], based on European insights, reveals that drivers in France have a tendency to enlarge their search radius the longer they spend looking for an available parking. When the search time for a vacant parking slot exceeds 15 minutes, the search radius becomes more and more significant and can extend beyond 500m, e.g., in the city of Lyon an average of 550m for a searching time of 45 minutes. Moreover, the authors conclude that vehicles in search for parking contribute to traffic congestion:

1. https://laexpresspark.org
2. https://europeanparking.eu
3. https://chinaparking.org

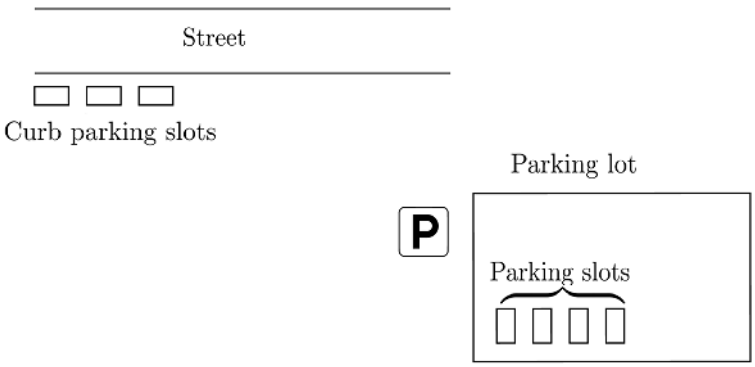

Figure 1.1 – Different types of parking places

between 5% and 10% of the overall city traffic and up to 60% in small streets. From these data, we can observe, and the literature confirms it, that most on-street related research papers were based on case studies from the USA, mainly California. On the other hand, researchers from other countries directed more attention to parking lots.

In terms of time spent, a study has estimated that on average, a driver in the UK will spend 106 days of his life just looking for parking [4]. In France a similar effect has been reported in [37], see Table 1.II.

	Total time lost	Lost time per hectare
Lyon district Presquîle	434h	14h 14min
Grenoble district Vaucanson	157h	6h 21min
Paris district Commerce	462h	9h 57 min
Paris district Saint-Germain	294h	13h 40 min

Table 1.II – Cumulative average lost time per day in several French districts, [37].

From these data it would appear that there are not enough parking slots to accommodate all the vehicles, but in fact, this may not be the case. Several studies show that in most cities there are sufficient parking slots for all vehicles and emphasize the negative environmental impact of constructing more parking [14, 23, 99]. Moreover, they underline the disutility of existing facilities by examining drivers behavior [104]. In general,

4. https://telegraph.co.uk/motoring/news/10082461/motorists-spend-106-days-looking-for-parking-spots.html

it is estimated that in the USA the parking to vehicle ratio is 8:1, being that for each vehicle there are eight parking slots. More precisely, parking coverage in the central business districts takes 31% in San Francisco, 81% in Los Angeles and 18% in New York of the land use. In other countries similar studies were conducted, for example 76% in Melbourne London 16%, and Tokyo 7%, see Evenepoel *et al.* [32].

During the years different approaches were developed to address the unorganized search for vacant parking spots. An obvious incentive is parking pricing. In other words, drivers would be more drawn to parking slots if they were cheaper. This approach gave rise to the parking pricing problem and produced a significant number of papers during the years. Another way to address the problem would be to reserve in advance a spot. Reserving a parking is particularly useful at airports, where the commuters usually know well in advance for how long they will be parked. An important factor of how the problem is addressed is based on current technologies. With all the available data and connectivity, the modern approaches are based on the internet of things and smart cities. This does not limit a combinatorial optimization approach, quite the opposite. With new embedded vehicular technologies and more available data, authors are able to formulate optimization models with greater precision than ever before, making the efficient search for a parking one of the central themes in many areas of research. Before addressing research papers, we take a brief look at some existing commercial systems that are deployed to assist drivers in finding a parking space.

Existing commercial solutions

Many cities have introduced some strategies to tackle the problem of finding a parking slot. One of the most frequent is the parking guidance and information (PGI), which is displayed on roads and continuously updates neighboring parking availability. Moreover, some indoor parking zones include adaptive lighting sensors, as well as parking space led indicators and indoor positioning system (IPS). Public parking lots with transport connections to city centers called Park-and-ride (P+R) are also present in most cities. Dating back from the 1930s, P+R facilities present incentives to commuters by providing bus, train or carpool to their destination, while leaving their vehicles in a less congested zone, where parking is easier to find [81].

There are also a variety of start-up applications that offer drivers guidance in order

to ensure an available lot in the vicinity of their destination. For instance, Parkopedia [5] keeps its content up to date via its users, who get credits for every entry (update) they make. Smartphone location feature enables apps to locate the nearest parking (mainly public garages). If the driver wishes, he can book a place at that parking via this application. An example of a successful project is the ZenPark mobile application [6], which embodies these characteristics and then estimates the quantity of saved CO_2, as a mark of environmental benefit. The local authorities of Lille (France) have developed a smart phone application containing useful real-time information called MEL [7]. Among other options the users can check the availability of most parking spaces (also public garages) in the city. However, guiding options are not included.

Modern car infotainment (navigation) systems also provide a list of parking lots when drivers choose their destination. One such example is the BMW iDrive system called ParkNow [8]. This onboard application allows drivers to settle their parking fee directly from their dashboard. However, it is focused only on reserving a parking in advance and does not take into consideration other vehicles.

The revenue that parking generates is especially interesting at airports, where commuters leave their vehicles for a longer period. This phenomenon became more present with low-cost carriers and increase non-aeronautical revenues for the airports and has grown to be an important source for airport revenue, see Yokomi *et al.* [116]. Therefore, most airports offer the option of booking a space for a specific period via their web site or the airport mobile application.

We can see that the parking-related problems are mostly still open topics and that many attempts have been made to improve the situation. In some cases, such as in the aeronautical, i.e., in airports, the parking system is more efficient that in urban areas. Moreover, we can observe that placing parking availability information (PGI, IPS, mobile applications, etc.) does not necessarily yield a significant traffic improvement.

Emerging new types of vehicles will be mostly electric. Electric vehicles (EV) have advantages over fossil-fueled vehicles, but they share the fact that they occupy, more or less, the same space as them and they too must be parked somewhere. Moreover,

5. https://parkopedia.com
6. https://zenpark.com
7. https://lillemetropole.fr/en/mel.html
8. https://de.park-now.com/en/2017/03/07/unique-integration-of-parking-payments-in-bmw-cars/

EVs have their own design constraints that introduce new challenges such as: finding a available charging station, battery autonomy left to reach it, shared usage consequences, etc. Therefore the problem of finding a good parking will remain a topic of research even when fossil-fuelled vehicles become a thing of the past. The following subsection presents some ideas that appeared in the literature about how to manage EVs parking.

Electric and autonomous vehicles

From what can be observed today, it appears that electric and autonomous vehicles (AV) will replace the petrol-fueled cars in a not so distant future. This is why in this subsection we briefly survey papers which discuss the parking issue of EVs and AVs and their specific needs. A simple search on Science Direct with the key words: "electric vehicle" and "parking" resulted on more than 3,500 papers published since 2015 and 1,633 results for the period between 2010 and 2015. This significant growth in the number of publications on the topic further illustrates the urgency of finding a good solution of parking EVs, before they fully enter the market.

The main disadvantage of EVs is the same since their creation in the nineteenth century: low range (Cavadas *et al.* [15]). Moreover, their charging times are also significant when comparing with combustion vehicles. This is one of the main reasons why their parking is important: they can be recharged while parked.

Jannati & Nazarpour [52] consider a photovoltaic-based intelligent electric vehicles parking lot to satisfy both environmental and economic issues of parking lots for EVs. In order to solve this problem, they formulate a bi-objective optimization framework. A similar topic is addressed in [82], where the authors discuss the technical, environmental and financial issues constraining the development of solar parking lots. Ioakimidis *et al.* [50] focus on the power consumption of EVs on a parking lot and optimize the charging/discharging of an electric vehicle on a parking lot. A study on the impact of EV parking in four different cities (Lisbon, Madrid, Minneapolis and Manhattan) was published in [34], where the authors evaluate the investment and benefit trade-offs and report their findings.

Autonomous vehicles don't need to be parked close to their destination, or even to park at all. Instead, AVs can seek out free on-street parking, return home, or cruise (circle

around), see Millard-Ball [65]. In this paper, an autonomous vehicle parking problem is formulated that simulates a game theoretic model based on data from San Francisco. The author of this paper considers three potential strategies for AVs once the driver exits it: free on-street parking, return home or cruising. The simulation experiments indicate that AVs could more than double vehicle travel to, from and within dense urban cores. More specifically, instead of keeping the AVs stationary in a parking lot, they could be used to double the total routes made in urban areas.

From this short review we can conclude that the parking related problems have been present for almost one century and will remain a topic of research in the years to come. The following sections will present some of the techniques applied to solve the parking problem. We start with the oldest technique: parking pricing. Data availability and network infrastructure make it possible to formulate more elaborate models, thus Section 1.3 surveys papers that make use of such features. These features also allow to formulate a combinatorial optimization model. A detailed survey on combinatorial optimization formulations will be presented in Chapter 2, but Section 1.4 briefly introduces some ideas developed when modeling the parking problem as a combinatorial optimization problem.

1.2 Parking pricing

A straightforward manner of simplifying parking search and alleviating parking caused traffic congestion is to set pricing incentives. More precisely, cheaper parking rates would theoretically attract more drivers, hence enabling a better usage of existing facilities. This was one of the first ideas when considering the parking problem. Already in 1928 the parking-meter was patented in the USA (patent number US1853103A) by Joseph Weisinger and a similar patent two years earlier called *Means for regulating the use of street parking space* by Charles C. Doyle (patent number US1752071A). Both patents were focused on curbside parking, but since then the use of parking lots is recommended and curbside parking is considered less favorable ([14, 97–99, 104]). This is why the transportation literature comprises many papers that offer solutions on how to set a parking tariff in order to better organize traffic and use more efficiently existing parking capacities. For example, McShane & Meyer [64] classify parking management

strategies according to the control exerted over the amount of parking supply. They conclude that the correlation between parking and urban objectives is strong and base their observations on a Baltimore case study. One of the main advantages of parking pricing is that it is not dependent on the technologies, i.e., the pricing strategies do not need do be determined by an advanced network, or other sophisticated infrastructure. For more insight on the economics of parking pricing we refer the reader to Anderson & de Palma [3]. In this subsection, we briefly survey papers which address the parking pricing problem (PPP) from an optimization point of view.

D'Acierno *et al.* [22] underline the simplicity of implementing different parking pricing and formalize several optimization models. Their goal was to rebalance the modal split between private car and transit systems in urban areas by setting different fees through road pricing and/or parking pricing strategies. Curbside parking still remains the most common way to park for drivers in the USA. Hence, Millard-Ball *et al.* [66] assess the curbside parking pricing in San Francisco. They evaluate the relationship between occupancy rules and metrics of direct policy interest, such as the probability of finding a parking space and the amount of cruising with the San Francisco performance goal of 60-80% occupancy of its on-street parking. More recently, the curbside parking effects on a case study in Rome was presented in [85].

The parking price can be modified during the day to further adapt to the demand. In other words, the pricing does not need to be fixed, but can be completely variable. This is why authors have formulated several dynamic versions of the PPP. A more recent example of such a formulation is presented by Zheng & Geroliminis [117], where the authors point out the impact of parking limitation on mobility and focus on modeling multimodal traffic with the treatment on parking, in order to implement the dynamic parking pricing strategies. In [63], the authors take advantage of technological possibilities in order to maximize parking space utilization. They formulate a dynamic non-cooperative bilevel model and claim that if it was applied, it could potentially eliminate the cruising for parking effect.

The main disadvantage of focusing on parking pricing is that prices do not guarantee better parking/vehicle distribution, and if we take into consideration all the parking lots, in practice their prices must be significantly different in order to provide sufficient incentives for drivers. This can lead to complex optimization models, which do not

necessarily lead to the desired effect. Moreover, the problem is frequently modeled to maximize revenue for the parking agencies and focuses less on the impact on traffic. One way to remove such disadvantages is to assume more control over the traffic and by incorporating other information, such as the locations of the vehicles, the parking availability and a network that collects and disseminates significant data. These data can help to assemble a different form of parking related problems, as will be demonstrated in the next section.

1.3 Smart city parking

In contrast to the PPP, where there can be very little technological requirements, in the era of internet of things (IoT) and under the umbrella of smart cities, exactly the opposite premise dominates: a strong network/device infrastructure. Unsurprisingly, most of the recent papers dealing with parking related problems fall under this predicament. These papers do not focus on the combinatorial aspect of the problem, but rather investigate information collection, system deployment and service dissemination, see Figure 1.2 from Lin *et al.* [61]. This section analyzes and synthesises some influential papers on this topic.

Delot *et al.* [27] examine the fairness of parking allocation in vehicular networks. As in vehicular networks it is less clear which slot would be globally best, the authors propose dissemination protocols with an *encounter probability parameter*. It estimates the likelihood that a vehicle will meet a certain event and shares the available information according to the encounter probability parameter while Cenerario *et al.* [16] focus more on disseminating and exchanging data about relevant events. In a more recent paper [108], the other aspects of sharing data in decentralized vehicular networks, such as congestion and hazardous road situations are investigated. Delot *et al.* [25] proposed a reservation protocol within a vehicle ad hoc network (VANET) system called VESPA, previously proposed in [24], by managing important parking notification to drivers, and further analyze the information dissemination in [26].

VANETs are decentralized networks, i.e., networks where information is partially available to users, depending on their location. On the other hand, we can assume that all the data are available to some central server, i.e., a centralized system. This assump-

tion produces different approaches to the parking problem, such as in Dsouza *et al.* [31] or Farag *et al.* [33]. More recently, Yang & Lam [115] proposed an intelligent parking information systems in a smart city environment which improves the efficiency of disseminating real time parking vacancy information. In Tang *et al.* [102] we see both a combinatorial formulation and the use of a significant IoT infrastructure. The authors formulate a simple assignment problem for attributing parking spots to vehicles, but mainly focus on data dissemination via fog computing.

Another good example of the smart city approach to parking management can be found in Xu *et al.* [113]. In this paper, the authors focus on providing an efficient software that they call the *phonepark system*. This systems takes into account the historical profile of parking and a mobile device that detects the status of the vehicle: parked or not. The phonepark system then automatically detects parking and deparking activities. The authors focus on network requirements such as the connectivity, transportation monitoring, paying via the smart device, accelerometers, etc. They then develop adequate algorithms to produce a parking availability estimation and evaluate them via simulation over an area of San Francisco.

The use of modern equipment is a natural way to address the parking problem. However, allocating parking slots was never truly defined in the literature as a combinatorial problem, although it distinctly is one. Furthermore, in practice, an efficient and realistic reservation system cannot be applied to any parking facility. In this thesis we do not reserve a parking to vehicles, but only allocate it a parking facility where there should be an available parking slot at their time of arrival, see Chapter 4.

So far we have discussed how parking pricing and how new connectivity features could be used to improve parking management. However, we have not yet addressed the problem of directly assigning parking to vehicles. Therefore, an efficient parking allocation mechanism, which could also include reservation systems, should be defined and well formulated. The following section introduces the parking allocation problem as a combinatorial optimization problem and briefly surveys papers that state it as such. Some papers have lower infrastructural requirements, e.g., [7, 36], while others heavily depend on external data, e.g., [102].

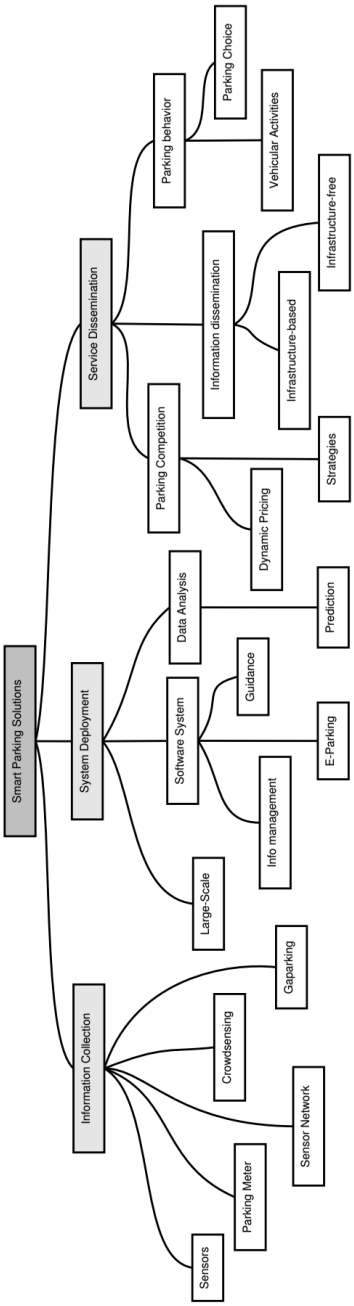

Figure 1.2 – Categories of smart parking proposed by Lin *et al.* [61]

1.4 Combinatorial optimization formulations

Allocating parking facilities to vehicles has been studied by the OR community for more than 40 years. One of the first surveys appeared in Florian & Los [36]. Although the problem of allocating cars to parking facilities is clearly combinatorial, there is no uniform MP model for it, and no standardized formulation. This is probably due to the very large number of variables and parameters that would have to be approximated and would lead to ambiguous results. This is why several authors have proposed various ways of defining the problem at hand. Unlike the PPP or smart cities, this section takes interest only in assigning parking to vehicles. Different authors opted for different formulations of the problem, but if the goal is to assign parking to a set of vehicles, then we will refer to this problem as the *parking allocation problem* (PAP) from now on.

For example, Ayala *et al.* [7] opt for a game-theoretic approach and model the PAP in a similar way as the *stable marriage problem*, and name it the parking slot assignment game problem. The stable marriage problem consists of finding a matching between two sets of elements of equal cardinality, given an ordering of preferences for each element. In Verroios *et al.* [110], the authors propose a traveling salesman problem (TSP) variant model: the time-varying TSP (TVTSP). The authors formulate it as a TVTSP because they identify the parking and destinations as points to be visited, which also depend on the time at which the parking slots become available. To solve the problem they propose several algorithms based on top-k and k-medoids methods, to produce a subset of parking spaces and group vehicles into clusters to improve the algorithm's efficiency. The top-k and k-medoids are types of k-medoids clustering algorithms (also called partitioning around medoid) that group items into clusters with minimal dissimilarities ([92]).

Ratli [90] focuses on the paths vehicles should take to reach their parking and destination. He thus invests on determining the shortest path with multiple objectives. Moreover, the parking assignment problem is modeled as a variant of the bi-objective assignment problem, minimizing the waiting time and traveled distance. In order to solve it efficiently, the author attributes weights to distances and waiting times to aggregate into a single-objective model. He then proceeds to develop a hybrid genetic algorithm heuristic and estimates the parameters of the dynamic variables by means of various learning techniques.

If we take into account only goods distribution vehicles, the problem can be seen as a variant of the vehicle routing problem with time windows (VRPTW) as suggested in Roca-Riu *et al.* [91]. The authors of this paper consider the limited parking availability for distribution vehicles and the strict timetable they have to respect. Several mixed integer formulations are discussed and solved by a standard MP solver. When delivery behavior is included, a different model can be stated, as in [2]. In [39] a dynamic scenario of resource allocation is considered. In that paper, each driver sets two values (upper bounds) on the parking cost and on the walking distance between the parking and the driver's destination he is willing to accept. Their system then reserves a spot for the vehicle by solving a mixed integer programming (MIP) formulation at each decision moment.

It can be observed that despite the fact that the problem is highly dynamic and combinatorial, few papers address both dimensions at the same time. In recent years the number of papers dealing with dynamic combinatorial problems in the field of transportation has grown rapidly (see [11, 19, 69, 86, 89]). The aim of dynamic formulations is to take into account the updated inputs and reevaluate current decisions. At any instant, there can be new requests, cancellations, failures or other unpredictable circumstances which would render a static model inapplicable. This is especially true when assigning parking to vehicles.

Most papers cited in this section propose some kind of optimization model to formulate the PAP. Some studies refer to their formulations as dynamic, if the model is time-dependent. In other words, if the time is included within the formulation. But it is clear that the main component of a highly dynamic problem is its continuous input updates. Therefore there should be a way to formulate a combinatorial problem capable of tackling dynamic updates. However, we see that the OR literature falls short in this respect. There is no clear consensus of how to solve or even formulate these types of problems. Chapter 2 will present how some well-know optimization problems were modeled to incorporate dynamic changes. From there, we will introduce how those methods can be applied to the PAP.

1.5 Conclusion

This chapter showed that parking related problems date back to the beginning of the 20^{th} century. We also notice that there are much more parking surface than would be initially expected, for example 81% of central business land area in Los Angeles. Further, according to some studies, cruising for parking is one of the main causes of traffic congestion: up to 60% of vehicles in traffic are cruising for parking. On a cumulative level several hundreds of hours are lost daily in some districts just searching for a vacant parking spot. With the emergence of EVs and AVs we further see that the issue of parking management and an intelligent solution will be even more important, if not crucial for a good traffic flow. Nonetheless, it seems that problems caused by parking have been rising year by year.

The first attempts to coordinate vehicles to different parking spots were undertaken by introducing parking pricing policies. However, these policies do not directly assign a parking to vehicles, rather they design the prices that could potentially improve traffic and focus mainly on increasing revenue. Some results further indicate that if the prices of parking are not significantly different, the cost of fuel will surpass the potential price benefits of parking pricing. Note that some of existing PPP models are very complex, and difficult to solve quickly.

Nowadays, with modern connectivity and abundance of data, it is realistic to assume that vehicles positions and destinations are known in real time. The infrastructure that enabled this technological leap allowed some authors to tackle the parking-related problems in a different fashion, within the concept of smart cities. Under the predicament of smart cities, authors focus on the flow of information and mainly try to reserve in advance a parking spot to a vehicle.

There have been many attempts to model the PAP as a combinatorial optimization problem. Some authors opt for game-theoretic approaches (e.g., [7]), others for different variants of well-established MP models (e.g., [75, 91, 110]). Most of recent papers on this topic make use of new technologies and suppose that vehicles are inter-connected ([61]). However, we observe a void in the literature when discussing dynamic updates in traffic and its influence when assigning parking to vehicles.

In this thesis we address the problem head-on. More precisely, we make use of only essential infrastructure requirements and constraints to first formulate a time-dependent MP model and then configure a mechanism in which it can be dynamically deployed to solve the model in near real time. Therefore, in the following chapter we survey the state of the art primarily on dynamic formulations and on their potential application to the PAP. Moreover, we focus on centralized systems, i.e., systems where the data of a set of vehicles (not necessarily all the vehicles in traffic) are available to a central server.

CHAPTER 2

THE PARKING ALLOCATION PROBLEM: A STATE OF THE ART

In Chapter 1 we saw a variety of ways to tackle parking related problems. In this chapter we focus on the parking allocation problem (PAP). More precisely, the problem of assigning a parking to vehicles. This is not a well-established problem in the literature, and thus it has no standard formulation. Moreover, at its core the PAP is dynamic, i.e., its input it continuously updated, and therefore a standard static model would not be a good representation of the problem itself. However, over the years researchers developed several ways to formulate the PAP. Some models are dynamic, but most are static. The PAP has no definitive model, but it can be considered as a variant of the assignment problem. The classical assignment problem is static, but some papers also investigate its dynamic variant, e.g., [100]. This is why this chapter starts with several variations of the assignment problem that could be used to model the PAP in a static setup. The chapter then proceeds with dynamic versions of some other well-established optimization problems. Once an overview of existing dynamic formulations is presented we turn our attention to the PAP. We first review the literature and classify the papers with an existing taxonomy from the dynamic vehicle routing problem. We then proceed by proposing a classification specific to the PAP.

Chapter outline As previously mentioned, the literature on the parking assignment offers two main threads: static and dynamic. But before addressing the parking assignment problem, we first survey a number of various assignment problems (AP) in Section 2.1. Section 2.2 sheds light on how some authors tackled different variations and extensions of the AP in a dynamic setup. The next section offers an extensive literature survey on the various parking assignment problems, both dynamic and static. Section 2.4 takes interest in the methods, exact and heuristic, used to solve the diverse parking assignment problems. This survey allowed us to propose a classification of existing formulations and solution methods in the Section 2.5. The chapter is concluded in Section 2.6.

2.1 Assignment problems

The main goal of any assignment problem is to match elements from two sets, respecting some constraints while minimizing (maximizing) some cost (profit) function. In this context, we can observe that allocating parking to vehicles has to contain some kind of assignment problem in it. Therefore, we first present some of the more influential variations of the AP, before proceeding to the PAP. Assignment problems are among the first problems to have been studied in OR, where the classical AP has been around for more than 60 years. A survey of variations of the AP and on the behalf of its 50^{th} anniversary was published by Pentico [84], from which we selected a few examples that will be useful for the PAP.

Classical assignment problem

More than 60 years after Kuhn's seminal paper regarding the AP, it still remains a topic of research in its different forms. A reprint of his paper was published in Naval Research Logistics in honor of its 50^{th} anniversary (Kuhn [54]).

When assigning tasks to agents, jobs to machines, jobs to workers, workers to machines, or parking to vehicles, etc., we encounter some kind of AP within the model. The classical AP is usually described as that of finding a one-to-one matching between n tasks (jobs, workers, parking) and n agents (machines, workers, vehicles) minimizing the total cost of the assignments:

$$\text{Minimize} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{2.1}$$

subject to:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,\ldots,n \tag{2.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,\ldots,n \tag{2.3}$$

$$x_{ij} \in \{0,1\} \qquad i = 1,\ldots,n, j = 1,\ldots,n, \tag{2.4}$$

where $x_{ij} = 1$ if and only if agent i is assigned to task j and c_{ij} represents the cost of

assigning agent i to task j, or in our context the cost of vehicle i reaching parking j. Note that the classic AP is mathematically identical to the weighted bipartite matching problem from graph theory. This model is one of the first problems to be provided with a polynomial algorithm for solving it. Furthermore, the restriction of $x_{ij} \in \{0, 1\}$ is unnecessary because the constraint matrix (2.2)-(2.3) is totally unimodular (TU) and thus always produces integer solutions, see Burkard *et al.* [12], Geoffrion [41] or Schrijver [94].

Bottleneck assignment problem

The bottleneck AP is a variant of the AP, proposed by Gross in 1959 [43], where the objective is to minimize the maximum of the agents costs :

$$\text{Minimize} \max_{i,j=1,\ldots,n} c_{ij}x_{ij}. \tag{2.5}$$

subject to the same constraints (2.2)-(2.4). The main difference between the AP and the bottleneck AP is that the linear relaxation of the decision variable x does not necessarily lead to integer solutions. Most variations of the AP have their bottleneck versions. In other words, if the objective is the maximal processing time, cost etc., then the word bottleneck is added to the name of the problem. When considering the PAP, the bottleneck AP can represent the case when we want to increase the fairness of the parking allocation by minimizing the maximal travelling time.

Generalized assignment problem

The basic variant of the AP that allows an agent to be assigned to multiple tasks is called the generalized assignment problem (GAP). Each task will be assigned to one agent, but an agent may be assigned more than one task, taking into consideration the agent's capacity to perform those tasks. The GAP can also be viewed as the problem of scheduling parallel machines with costs [96]. There are m jobs and n machines. Each job is to be processed by exactly one machine, processing job j on machine i requires a_{ij} amount of time and incurs a cost of c_{ij}. Each machine i is available for b_i time units, and

the objective is to minimize the total cost. The GAP is generally formulated as follows:

$$\text{Minimize} \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \tag{2.6}$$

subject to:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, \ldots, m \tag{2.7}$$

$$\sum_{j=1}^{m} a_{ij} x_{ij} \leq b_i \qquad i = 1, \ldots, n \tag{2.8}$$

$$x_{ij} \in \{0,1\} \qquad i = 1, \ldots, n, j = 1, \ldots, m. \tag{2.9}$$

The GAP is NP-hard, but it is highly sensitive to its input. More precisely, although NP-hard, it can be solved optimally with commercial solvers for certain inputs. For example, the instances from the OR library Beasley [8] are all easily solved, but the instances proposed by Yagiura *et al.* [114] show that heuristics are preferable. A similar barrier for exact methods can be the scale of the task/resources set, where again heuristics represent a better solution method, see Mitrović-Minić & Punnen [68].

Note that in the context of assigning parking to vehicles, we can consider each parking slot individually, or all the available slots of a given parking lot. If a parking lot is considered, then we would encounter some kind of GAP because of the limited capacity of each parking facility. This would result by some type of capacity constraint in the PAP model, such as (2.8). Whereas if each parking slot would be considered separately, the constraint (2.8) would not exist.

Quadratic assignment problem

A simple formulation of a problem does not guarantee a simple method for solving it. This is the case with the quadratic AP, which is one of the most challenging combinatorial optimization problems. In this problem we want to assign n facilities to n locations. The cost of assigning facilities i to k, is the product of the flow between i and k, denoted by a_{ik}, and the distance between the locations j and l, denoted by b_{jl}. The cost of placing new facilities at their respective locations, denoted by c_{ij} is also included

in the cost. The objective is to allocate each facility to a location such that the total cost is minimized Burkard *et al.* [12]. The quadratic AP can be stated as follows

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{ik}b_{jl}x_{ij}x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{2.10}$$

subject to:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,\ldots,n \tag{2.11}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,\ldots,n \tag{2.12}$$

$$x_{ij} \in \{0,1\} \qquad i = 1,\ldots,n, j = 1,\ldots,n, \tag{2.13}$$

where x_{ij} is equal to one if and only if facility i is assigned to location j, with the corresponding placing cost equal to c_{ij}. Applications of the quadratic AP include the following areas: scheduling, parallel and distributed computing, statistical analysis, sports, chemistry, archaeology, computer manufacturing, and transportation. An entire library of instances for the quadratic AP can be found in Burkard *et al.* [13]. The quadratic AP is so hard that it took researchers more than 30 years to solve exactly instances of size 36, see Taillard [101]. The quadratic AP contains the TSP, and is as such NP-hard.

Multi-dimensional assignment problems

In some cases of the AP, we can match the members between three, or more sets, instead of just two. For instance, the problem of matching jobs with workers and machines or one of assigning students and teachers to classes and time slots (the timetabling class of problems). A survey of examples of multi-dimensional assignment problems is presented in [84]. It includes the planar three-dimensional assignment problem, the axial three-dimensional assignment problem, the three-dimensional bottleneck assignment problem, the multi-level generalized assignment problem, etc. Most of the multi-dimensional APs include time, most frequently by assigning a task at a specific time. For example, the planar three-dimensional AP involves scheduling meetings between r vendors and s customers over t time periods ($r \geq s \geq t$). During each time period, every customer is to meet one vendor and each vendor is to meet at most one customer. The

problem is stated as follows:

$$\text{Minimize} \sum_{i=1}^{r} \sum_{j=1}^{s} \sum_{k=1}^{t} c_{ijk} x_{ijk}$$

subject to:

$$\begin{aligned}
\sum_{i=1}^{r} x_{ijk} &= 1 && \forall j,k \\
\sum_{j=1}^{s} x_{ijk} &\leq 1 && \forall j,k \\
\sum_{k=1}^{t} x_{ijk} &\leq 1 && \forall j,k \\
x_{ijk} &\in \{0,1\} && \forall i,j,k,
\end{aligned}$$

where c_{ijt} represents the cost of arranging such a meeting. If we consider vehicles as customers and parking slots as vendors, then the goal could be to assign a slot to a vehicle at some period. This rise of dimensionality usually causes the problem to be more difficult to solve, thus creating the need for efficient heuristics (see Gilbert & Hofstra [42], Laguna *et al.* [55]).

Multi-objective assignment problems

A multi-objective combinatorial optimization (MOCO) problem can be presented in the following form:

$$\text{“Minimize”} \ \{g(x) : x \in X\}, \tag{2.14}$$

where g is a vector function $g : X \to \mathbb{R}^n$, $g(x) = (g_1(x), \ldots, g_n(x)), n \geq 2$, and X is the feasible set. Note that, we used the term “*minimize*” for the optimization problem (2.14), because the optimal solution of (2.14) is relative. We consider only the case $X = \{0,1\}^p, p \geq 1$. A solution $x^* \in X$ of (2.14) is called *efficient* if $g_i(x^*) \leq g_i(x), i = 1, \ldots, n$ for any other $x \in X$, with at least one strict inequality. Let the space of all efficient so-

lutions of a problem Π be denoted by $E(\Pi)$. Denote the MOCO problem (2.14) by Π. Let Π_λ be a parametrized linear programming model of Π, i.e., $g_1, \ldots, g_n$ are real linear functions, $X \subseteq \mathbb{R}^n$ and $\lambda_i > 0,\ i = 1, \ldots, n$ are pondered weights, i.e., $\sum_{i=1}^{n} \lambda_i = 1$. Then the optimal solution x^*_λ of the single objective problem

$$\text{Minimize } g^\lambda(x) = \sum_{i=1}^{n} \lambda_i g_i(x), \tag{2.15}$$

subject to $x \in X$ is an efficient solution of Π. This property is referred in the literature as the *Geoffrion's theorem* [40]. The optimal solutions of Π_λ are called *supported efficient solutions* of Π. The set containing all supported efficient solutions of Π is denoted by $SE(\Pi)$, while solutions belonging to $E(\Pi) \setminus SE(\Pi)$ are called *non-supported efficient solutions*. A corollary of this property states that all the solutions of a multi-objective LP are supported, see Isermann [51].

Although the classical AP possesses the unimodularity property and thus its linear relaxation yields binary values, if the objective function in not scalar, it then looses its properties. More precisely, Geoffrion's theorem is not applicable because the AP feasible set X is not convex. In other words, if problem Π is an MOAP (2.14) then $E(\Pi) \setminus SE(\Pi) \neq \emptyset$. This is the case for almost any problem of class P (solvable in polynomial time) for vector objectives, see Ulungu & Teghem [109]. A simple example which illustrates this property of the multi-objective AP is presented in White [112].

In the case of the parking assignment, we can set the objectives to be the cost and the traveled time, for example. An objective can also be the waiting time, i.e., the time a vehicle should wait before a parking is allocated to it. However, in this thesis, we will focus only on mono-objective PAP models.

We can conclude that APs have a wide array of applications and variations. Some are easy to solve, while others represent some of the most challenging problems in OR. As such, both approximate and exact methods have been developed to solve them. Additionally, if the time element is incorporated into the model, we encounter multi-dimensional versions of the AP, which are usually more difficult to solve than standard two-dimensional variants of the AP. Moreover, if the objective is a vector function, the classical AP loses its unimodularity, i.e., its linear relaxation does not necessarily yield efficient solutions.

2.2 Dynamic optimization problems

The PAP is in practice a dynamic problem. Hence we first investigate the state of the art of some dynamic formulations in the literature before going into more detail about different versions (static and dynamic) of the PAP. In this section we take a closer look at papers that introduce the time element in some well-known problems, such as the AP and vehicle routing problem (VRP). As in Section 2.1, we start off with the dynamic version of the classical AP, then consider a dynamic version of the GAP. The AP can be considered as a special case of the VRP (see [89]), and thus a brief survey of the dynamic VRP (DVRP) and dynamic pickup and delivery problem (DPDP) are detailed in their respective subsections.

Including time into the problem does not make it necessarily dynamic. From now on we shall refer to a problem as dynamic if the input is revealed gradually over time and by consequence, the decision variables values change adequately. The formulations that include time, but do not include the option of changing their decisions will be regarded as *time-dependent* problems. For example, the TVTSP and VRPTW are time-depended problems, but not dynamic. Note that, a problem can be both dynamic and time-dependent. Problems which are sequentially solved as soon as new input is revealed, are also known as *online* problems. Note that, the term online and dynamic are not the same. All online problems are dynamic, but not all dynamic formulation are of the online type. The term *offline problem* is most often used to describe experiments in a controlled environment, e.g., a lab or a simulation ([87]). By contrast, online problems are those which are applied sequentially in the field. That does not mean that there are not sequential algorithms in offline settings. Before proceeding to dynamic versions of some well established problems, we first introduce some terms and notations specific to dynamic problems. This is especially important when attempting to classify existing papers of the PAP, presented in Section 2.5.

Stochastic optimization problems

To have a better understanding of dynamic problems, we will borrow the notation of stochastic optimization problems (SOP). In general, when a SOP is considered, it is assumed that the data is not fully available from the beginning, i.e., that there exists some

uncertainty surrounding the input. In many cases these uncertain data are regarded as future input, which have yet to be revealed to the decision maker. This point of view draws a strong parallel with the definition of a dynamic problem that we are using. The main difference between a SOP and a dynamic one is that the output of a SOP is a function and not a value (vector or scalar) as is expected of a dynamic problem. The resulting function (output) of a SOP is most frequently referred to as the *optimal* or *best policy*, and less frequently (depending on the community) the *decision rule*. Once the best policy is determined, concrete decision can be made. Therefore, we can consider that a dynamic problem is a specific type of SOP, where the solution can be a vector and not a function. In other words, a dynamic problem can be viewed as a SOP for a fixed policy.

The main point in common between dynamic problems and SOPs, and which is specially emphasised in SOPs, is the fact that the state of the problem changes as new input arrives. The state also depends on previously made decisions (solutions) and not only on raw input, e.g., number of vehicles, depot changes, traffic perturbation, etc. In other words, SOPs incorporate by default a transition of states over time, as well as updated input. The dynamic problems focus more on providing good solutions with the currently available data, from which the term online problem was coined. Online dynamic problems are specifically lack in incorporating state changes into their formulations even thought they rely on sequentially solving static subproblems, as shall be detailed later in this section. Therefore, it can be said that SOPs are more useful as strategic (long-term) or tactic (mid-term) landmarks, while dynamic problems can be considered more as a tool for operational (short-term) decision making.

It is obvious that most real-world problems are stochastic and dynamic, meaning that (future) input can rarely be absolutely accurate (probabilistic) and one solution for all time will just not cut it (dynamic), i.e., solutions may require adjusting over time. This is the case for most engineering sciences, and is not unique to transportation sciences or OR. Therefore, different communities of researchers have addressed similar difficulties in various ways: Markov decision process, stochastic programming, online algorithms, robust optimization, optimal control, simulation optimization, etc., see Powell [87]. In order to precisely define the PAP and thus suitably solve it, we first briefly present how research from other domains tackled dynamic problems. More details about specific

techniques in dynamic problems will be presented in Chapter 4, while solution methods (strategies) will be addressed in Section 2.4. Note that, online algorithms refer to the methods applied to solve a dynamic problem, while an online problem (or data-driven) is the one where data are received as time progresses.

Online algorithms According to Powell [87], the most common approach to solving SOPs in transportation problems are online algorithms. In online algorithms, little or no knowledge about the future is assumed and (static) problems are solved sequentially as new data arrives. Online algorithms originated from the need to provide decision for robots or devices with limited energy resources, thus potentially producing myopic policies. To moderate these myopic effects, a research domain known as *competitive analysis* was established which in turn develops bounds on the performance compared to a far-seeing policy. However, online algorithms, if sufficiently efficient, can adapt to changes and can produce good short and mid-term solutions. This is particularly the case where real-time solutions are required, and where computing the best policy would require a considerable amount of time.

Markov decision process A Markov process represents a stochastic process that satisfies the Markov property, also called *memorylessness*. It is a stochastic model that describes a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. In other words, a process satisfies the Markov property when a prediction for the future is based only on its current state, i.e., its future and past states are independent. A Markov decision process (MDP) is not to be mistaken for a Markov chain. The MDP is a discrete time stochastic control process. It provides a mathematical framework for modeling decision making in cases where outcomes are partly random and partly under the control of a decision maker. Unlike the Markov chain, which originated in probability theory, the MDP has its roots in dynamic programming, see Bellman & Dreyfus [9].

A Markov chain is a type of Markov process either a discrete state space or a discrete index set (often representing time), but the precise definition of a Markov chain varies. In Asmussen [4] the Markov chain is defined as a Markov process in either discrete or continuous time with a countable state space. It is also common to define a Markov

chain as having discrete time in either countable or continuous state space.

Dynamic assignment problem

Spivey & Powell [100] investigated an interesting fact: although the AP is a useful and well established problem, its main application would be in a dynamic setup. More precisely, a dynamic setup represents the scenario where the task-resource matching is repeated over time, questioning previous decision with the newly available information. This simple premise completely changes the nature of the classical AP, similarly to the changes of switching to a vector objective function, as in the case of the MOAP in Section 2.1.

The authors rightfully conclude that little attention has been devoted to explicitly extending the classical (static) AP into a dynamic setting and formulate the dynamic AP (DAP). They propose a Markovian exogenous information process in which the heart of the decision process lies the classical AP. The Markov chain process was included mainly to encompass (simulate) data (tasks and resources) which are not yet visible. By doing so, the authors can anticipate future data and try to avoid myopic traps. More precisely, avoiding to assign tasks to resources which will be significantly suboptimal in the future. They propose an adaptive learning algorithm relying on Markov chain process that provides non-myopic behavior. However, the authors' experimentations show that the adaptive models will outperform myopic models with some advance information, but with sufficient advance information the myopic model actually outperforms the adaptive model. Moreover, they conclude that creating the right balance of completeness and simplicity is the appropriate way to approach the DAP, and that it could be generalized to other problems.

Another algorithmic strategy for the DAP can be based on multistage linear programs. These techniques can be divided between scenario methods that explicitly enumerate the space of possible outcomes: those based on Benders decomposition, and those that use other classes of approximations for the recourse function. Scenario methods require enumerating a set of outcomes and explicitly solving a large-scale program. Aside from the challenge of enumerating the space of potential scenarios, this approach destroys the natural integrality of the dynamic assignment problem, e.g., Laporte & Louveaux [59], Louveaux & van der Vlerk [62].

Dynamic generalized assignment problem

Kogan & Shtub [53] have proposed a formulation called dynamic GAP in 1997 in which each task has a due date. This problem is formulated as a continuous-time optimal control model. A combinatorial optimization problem, also called the dynamic GAP (DGAP), was proposed in Moccia *et al.* [80]. In this paper, the authors expand the GAP by introducing a discretized time horizon and by associating a starting and a finishing time to each task. The DGAP objective is to find a minimum cost assignment of tasks to facilities for each period of the planning horizon. The DGAP is called dynamic because the reallocations of tasks during this planning horizon are taken into consideration within the model. However, this does not satisfy our definition of a dynamic formulation because the decision will be determined by a single input, and will not change over time.

The authors propose two formulations: the first is a mixed integer programming (MIP) DGAP formulation, and the second is based on the origin-destination integer multicommodity flow problem (ODIMCFP). The authors then prove the DGAP to be NP-hard and that the ODIMCFP node–arc formulation dominates the DGAP formulation. To solve the DGAP they developed three column generation-based algorithms.

The DGAP can find applications in the management of warehouses and storage yards, where a task can be regarded as a group of items to be stored and a facility represents a storage area or position. Therefore, allowing dynamic reallocations often leads to a significant improvement in space utilization. Reallocations can also be motivated by operational considerations. For example, areas closer to arrival and departure gates are often used for temporary storage to accelerate loading and unloading operations. In their paper [80] the authors have adapted the DGAP to the berth allocation problem based on Cordeau *et al.* [20].

Dynamic vehicle routing problem

Like the AP, the VRP has marked its 50^{th} anniversary in 2009 (Laporte [58]). Over the years many different variations of the VRP were proposed, see [21, 60]. The number of methods for solving them is almost equally well studied, see [18, 56, 57]. Unlike the simple AP, the VRP has a significant body of literature of its dynamic counterpart, e.g., Gendreau *et al.* [38], Hvattum *et al.* [48, 49], Wen *et al.* [111]. Psaraftis [88]

was one of the first to attempt to explicitly solve a deterministic, time-dependent version (dial-a-ride problem) of the vehicle routing problem by means of dynamic programming. However, he encountered the well-known dynamic programming problem regarding dimensionality.

Although formulations for the dynamic VRP (DVRP) vary, the main idea remains the same: adapt the routes depending on the requests. To illustrate the DVRP we take the model as presented in Gendreau *et al.* [38]. The authors state the dynamic version of the VRP as a sequence of static VRPs which dynamically appear over time. The static model they consider is the VRPTW. For a given undirected and complete graph $G = (V,E)$, $V = \{v_0,\ldots,v_n\}$ denotes the vertex set where v_0 denotes the depot and E is the set of all the vertices. Every vertex i has a time window $\left[e_i, l_i\right]$ in which it has to be serviced (with e_0 being the earliest start time and l_0 the latest end time of each route). The distance matrix $D = (d_{ij})$ is assumed to be symmetric and the edges respect the triangle inequality. The fleet is comprised of m identical vehicles that need to service all nodes of V originating and terminating from the depot v_0 respecting the following:

— each vehicle services one route;
— each vertex is visited only once;
— the start time of each vehicle route is greater than or equal to e_0;
— the end time of each vehicle route is less than or equal to l_0;
— the time of beginning of service b_i at each vertex $v_i, i = 1,\ldots,n$ is greater than or equal to the earliest service time e_i, if the vehicle's arrival time t_i is less than e_i, a waiting time $w_i = (e_i - t_i)$ is incurred ([38]).

The MIP model they used was defined in Desrosiers *et al.* [28], with the difference that the time windows constraints are soft and are penalized in the objective function (2.16).

$$\text{Minimize} \sum_{i=1}^{m} d_i + \sum_{k=1}^{n} \alpha_k (t_k - l_k)^+, \quad s \in S. \tag{2.16}$$

The feasible set is denoted by S, d_i denotes the total distance traveled on route $i, i = 1,\ldots,m$, $y^+ = \max\{0,y\}$ and α_i is the lateness penalty coeficient associated to vertex i.

The requests are not known ahead, hence operating scenarios are included based on the following assumptions:

— requests must be received before a deadline to be serviced the same day;

— there are no cancellations, i.e., uncertainties come from a single source, new service requests;
— only the dispatcher has all the information, which he then transmits to the drivers;
— waiting time is included, i.e., drivers may wait to receive a route from the dispatcher.

The main difficulty of such a formulation is that the static VRP can be challenging to solve in its basic version. Solving it sequentially over time can cause compromises in the quality of the static solutions. Therefore, in [38] the authors developed a parallel tabu search (TS) heuristic for solving each cycle of the dynamic sequence.

A survey of thirty years of the DVRP was published in Psaraftis *et al.* [89]. The latter paper shall be used for our classification of the DPAP and to position our contributions within the literature in the Section 2.5 and for classifying existing solution methods of the PAP in Section 2.3.

Dynamic pickup and delivery problem

In this subsection we take a look in another well studied version of the VRP: the pickup and delivery problem (PDP). A survey of the static versions of the PDP can be found in Berbeglia *et al.* [10]. Furthermore, like the VRP the PDP has a significant number of papers dealing with its dynamic version. The surveys of the dynamic PDP that was used in this manuscript is Berbeglia *et al.* [11] and Psaraftis *et al.* [89]. However, we take the example of Mitrović-Minić *et al.* [69] to illustrate interesting features when PDP is dropped into a dynamic configuration.

In their paper [69] the authors consider a PDP with time windows (PDPTW), but within a dynamic setup. The PDPTW consists of determining a set of optimal routes for a fleet of vehicles while satisfying transportation requests: all requests must be served, time windows must be respected, each request must be served entirely by one vehicle (pairing constraint), and each pickup location has to be served before its corresponding delivery location (precedence constraint), with the goal of minimizing the total route length. The dynamic PDPTW arises when not all requests are known in advance. The main difficulty when solving PDPTWs is to make good short-term decisions without adverse long-term effects. For example, using a myopic policy in the short-term may remove the flexibility needed to make good long-term decisions. They propose the use of a

double-horizon when assigning new requests to vehicles and when scheduling the vehicles, where the double-horizon is based on the balance between the short-term and long-term decisions. They report that their results are superior to the standard *rolling horizon* which is often used. Furthermore, to evaluate the results of the dynamic PDPTW they introduce a parameter which they call the *value of information* that compares the result of the dynamic PDPTW, with the static PDPTW where all the information is available (overall PDPTW).

Discussion

In this section, we so far observed that a number of well-established problems have their dynamic analogs. We have also observed that there are many potential ways to tackle them, and that the way they are formulated and solved depends on the intentions of the authors. However, for all the previously presented dynamic problems, the authors seek to answer three questions:

(Q1) In which way is the input updated or anticipated?

(Q2) How to evaluate current decisions?

(Q3) How to change current decisions to avoid myopic traps?

The input can be updated by approximating it with a probability distribution (Poisson, negative exponential, see Section 2.3) or with a Markov chain process [100]. The second question, mostly relies on solving a combinatorial formulation, for the previously given input, e.g., [38, 69, 80, 100]. Once it has been solved, an evaluation phase intervenes in order to avoid myopia, i.e., adjusting the solution with certain anticipation of the upcoming input. We see that this last question raises the most challenges because it is difficult to evaluate, and hence adapt, a decision with low foresight. The following section addresses these same questions, but within the scope of the parking assignment problem.

2.3 Parking Assignment

Although no definitive formulation of the parking assignment problem exists, we can focus on some key facets. From our point of view, when considering the PAP the following three key facets should be addressed:

(i) a static formulation;

(ii) the methodology for solving the dynamic problem relying on the static formulation;

(iii) the decision criteria.

Zou *et al.* [118] provide specific answers to all three of the aforementioned facets.

(i) Static parking slot assignment: essentially an AP model for which the goal is to maximize social welfare by allocating slots to drivers. To do so, the parking manager needs to know the drivers valuations, or utility for parking, which is only known by the drivers themselves.

(ii) Methodology - dynamic parking assignment: an extension of the static case for which the authors divide the planning horizon into 20 periods, and solve the static PAP for each period. From the perspective of game theory, when the drivers approach the parking slot, they provide the parking manager with four values: the drivers arrival, latest waiting, and departure times and valuation (the amount a driver would be ready to pay for a given slot). The final results, would they be allocated or not to a parking slot depends on those four values.

(iii) Decision criteria - a mechanism design: a reverse game theory approach, in which the outcome of the game is predefined and the agents are expected to take their strategies to attain predefined outcomes. In their context, the mechanism designer correspond to the parking manager and the agents to the public parking manager and drivers respectively.

Without loss of generality, the first facet can be considered as a simple MP model which acts as the decision maker, i.e., the parking administrator. While the design mechanism acts as the overall guiding principle for any decision making, static or dynamic. The dynamic case in [118] is, in a way, also a static version of the PAP, but with more elaborate requirements such as arrival time, latest waiting, departure time, and slot valuation by drivers. The authors then primarily direct their attention to finding an assignment mechanism for the social optimum allocation outcome which can be achieved.

This paper underlines the core of the PAP: separating the decision process (parking allocation to vehicles) from the dynamic (overall) objective. More precisely, the allocation of parking facilities to vehicles and their overall configuration should be separated and considered apart. In previous sections, we saw that similar ideas were developed

to avoid myopic traps and incorporate the dynamic element, but Zou *et al.* [118] are among the first authors to propose a complete framework for tackling the PAP formulation. Other papers dealing with the PAP have focused more on solving a single configuration (facet) of a given PAP, as the following subsections will demonstrate. However, as will be shown in more detail in Section 2.5, Zou *et al.* [118] impose unrealistic assumptions and do not take into consideration the parking availability at arrival time. Other shortcomings can be found in the dynamic sequence which has little knowledge about previous or future requests. The main contribution of this paper lies their application of a reverse game theory methodology that aims to attain a predefined outcome. Mladenović *et al.* [77] was, to the best of our knowledge, the first paper to distinctly separate these three phases for a PAP, see Chapter 4.

AP-based PAP

The PAP can be considered as a variant of the assignment problem, such as in [1]. The authors of this study attribute parking lots to vehicles considering the parking time limit and their distance from the vehicles, also taking into consideration different parking prices and attributing a weight to each vehicle, based on distance and price. They call the problem the parking slot assignment problem for groups of drivers (PSAPG) and focus on alleviating the driver from following numerous parking availability boards. They directly assign parking lots to a set of interconnected vehicles. They develop several heuristics for solving the PSAPG and report a hybrid genetic-GRASP heuristic to be the best option. Tests are performed on 17 parking facilities in Tunis, with capacities ranging from 35 up to 1200 slots. The authors state that CPLEX was not able to solve instances with more than 30 vehicles. The input data is considered deterministic and static. This paper does not take into account the dynamic nature of the PAP, but instead takes a classical approach for solving a combinatorial problem: proposing a MP model, and then trying to improve the solver's results by developing an efficient heuristic.

In [75] the authors demonstrate that the 0-1 PAP model can be more efficiently stated. In the latter paper, they consider a set of interconnected vehicles with only capacity and allocation constraints. A residual capacity is introduced, which corresponds to the number of available parking spots at the arrival times of each vehicle. With this simple premise, they prove that their model possesses the integrality property. Therefore, unlike

in [1], CPLEX is capable of providing a solution for instances with 30 parking lots and 9,000 vehicles in less than a minute. In addition, a variable neighborhood search (VNS) heuristic was developed to allocate parking lots to up to 90,000 vehicles and 50 parking lots. Their formulation is also static and deterministic, but the authors propose a dynamic mechanism by sequentially solving the static PAP, similar to Gendreau *et al.* [38] for the DVRP.

In some countries the shared use of available parking spots is encouraged. This concept emerged from the idea of making a more efficient use of existing parking facilities [1] [2] [3]. It uses existing slots intended for parking cars when the owner is not using them. Parking availability for others stems from the fact that most parking spaces are only used part time by a particular driver or owner who lives in one location and works in another, thus the utilization and availability patterns follow predictable schedules. By making private parking space publicly available for rent, shared parking provides the owner with additional revenue and also helps alleviate the shortage of parking spaces. With this premise in mind Shao *et al.* [95] formulate an AP-based model as in [1, 75] for the core of the parking allocation decision process. The system then reserves parking spots to vehicles, keeping the option of rejecting some reservations, since the objective is to maximize the revenue of the parking owners. Therefore, their approach is also static and deterministic. The computational experiments favorably compares their model to the first-booked-first-served strategy. Further, when generating their random test instances, they assume that the expected parking duration is three hours with a negative exponential distribution. In addition, the arrival of requests follows a Poisson distribution. The planning horizon is set to be eight hours (between 09 h and 17 h) with up to 1,000 requests (vehicles) and 300 parking spots.

More recently, an assignment problem-based model was published in [102], proposing an MP model for the allocation of parking slots to vehicles. The model takes into account the price and waiting time and assigns parking lots to vehicles with the aim of minimizing a weighed cost. Their allocation constraints impose that every vehicle must find a parking slot. It then disseminates the solution through a vehicle ad hoc network (VANET), applying *fog computing* technique. Several greedy algorithms were devel-

1. https://roverparking.com
2. https://spot-park.com
3. https://parkcirca.com

oped for the proposed model and simulations were conducted to evaluate their efficiency over a section of Xuzhou City in China, covering six parking lots and up to 650 parking requests. The authors opted for a sequential dynamic setup, for which the decision process is based on the previously mentioned AP-based model. The entire approach can be viewed as online dynamic, and deterministic.

Other combinatorial PAP formulations

The combinatorial basis of the PAP does not need to be the AP, as this section will illustrate. As briefly mentioned in Section 1.4 other well-know MP models can be used as a tool for allocating parking lots to vehicles, such as the TSP. However, one of the main difficulties when considering a PAP is to guarantee an available parking slot to a vehicle at its actual arrival time. Therefore many authors emphasize the use of models in which a parking slot is reserved in advance.

Geng & Cassandras [39] propose a dynamic reservation system in which, if a driver is satisfied with the current parking allocation, he has the choice to reserve that spot. When a reservation is made, the driver still has an opportunity to obtain a better parking spot before reaching his allocated parking spot. If he is not satisfied with the assignment, he has to wait until the next decision point. The vehicles are stored into two unbounded capacity queues: the waiting and reserved queues. This reservation process is based on a sequence of mixed integer programming (MIP) problems solved over time at specific decision moments with the objective of minimizing a weighted cost/distance metric. These decision moments are determined in two ways: time-driven, and event-driven approaches. In other words, the decision moment in which the decision process (MIP) is evoked can be triggered by a predefined time step (periodically), or by an occurrence of a given event. The authors underline that their MIP formulation frequently cannot produce feasible solutions, due to the capacity constraints. They therefore relax that constraint by penalizing non-allocated vehicles. They evaluate their approach by simulating the MIP sequence on a small business district with four malls and 30 parking resources. Therefore, the approach is dynamic, as the decisions change at each decision moment. The input can be considered deterministic since the variables of the MIP are not stochastic, and the solution method is based on queuing theory, since the MIP itself is solved exactly with ILOG CPLEX solver.

Another example of a reservation system can be found in Teodorović & Lučić [103], who propose a system in which the drivers would reserve a parking, even before starting their trip. They propose an "intelligent" parking space inventory control system based on the combination of simulation, optimization techniques, and fuzzy logic that makes decisions to reject or accept new parking requests. The authors do not explicitly focus on assigning parking lots to a set of vehicles, but rather on maximizing the revenue by reserving in advance, similarly to booking a plane ticket. Still this paper incorporates well the methodology of the PAP: a combinatorial problem, solved in an online dynamic fashion.

Verroios *et al.* [110] also underline the fact that uncertain parking availability can present a challenge. They consider a network in which vehicles would keep useful information such as the average time periods needed to traverse road segments, the average number of spaces one may need to visit before an available spot is reached, and the average period of time that a parking space remains free. They then disseminate these data via a given network protocol. The quality of a route that visits reported parking places is assessed by a cost function. This function depends on the probability to actually locate an available spot to a vehicle. This is why they formulate the PAP as a variant of the TSP, i.e., the time varying TSP (TVTSP), where parking locations are considered as points to be visited. The authors introduce the *salesman method* which incorporates three approaches:

(i) the approach in which they determine the vehicle trajectory with the data available on-board at request time, that the authors call the *exact approach*;

(ii) the *clustering-based approach* in which instead of dealing with all candidate parking spots, they group all the currently on-board available spots according to their geographical locations;

(iii) the *live* or online *approach* where the parking spots updates are continuously received. Then the vehicles routes are recalibrated dynamically in order to reach their destination.

However, their implementation of the TVTSP and how the requests are updated and handled in the simulation are not discussed. It can be said that they propose a dynamic setup with deterministic input where the decision process is once again based on a combinatorial formulation.

Amer & Chow [2] make use of the known *A106 parking equilibrium model* with traffic flow behavior that explicitly measures cruising effects. The authors generalize the AI06 model to include trucks among other vehicles in traffic, which implies a delivery behavior. In order to include them, they introduced a number of new parameters into the A106 model. Street parking is considered and some vehicles can occupy more than one slot. Once new parameters and constraints were defined, they tested their approach on a case study over a part of the city of Toronto. In their paper the A106 serves as a protocol (black box) which produces a result and no static, or dynamic formulations are explicitly proposed.

2.4 Solution methods

In Chapter 1 and Section 2.3 we saw several types of combinatorial formulations of the PAP. In this section we focus more on the dynamic aspect of the PAP, following the DVRP taxonomy of [89], see Figure 2.1. We divide the survey of papers dealing with combinatorial formulations of the PAP by their way of modeling it. In other words, since no concrete PAP formulation exists, authors opt for different well-know models to describe the PAP.

In solving static optimization problems, two main types of solution methods exist: approximate or heuristic, and exact methods. This is the case for solving dynamic optimization problems. However, when solving the dynamic optimization problems as a sequence of static problems, this approximation induces an error. Therefore, with such an approach, the exact solution of the dynamic problem cannot be found, even if we solve exactly each static problem in sequence. In order to minimize the gap between these solutions, some authors have proposed various approaches, mostly by introducing additional decision variables between solving two consecutive problems, that were not considered in the static formulation. For example, in [118], to make the decision to accept or reject a vehicle's request, variables such as the drivers arrival, latest waiting, departure time and valuation are introduced after solving the static problem.

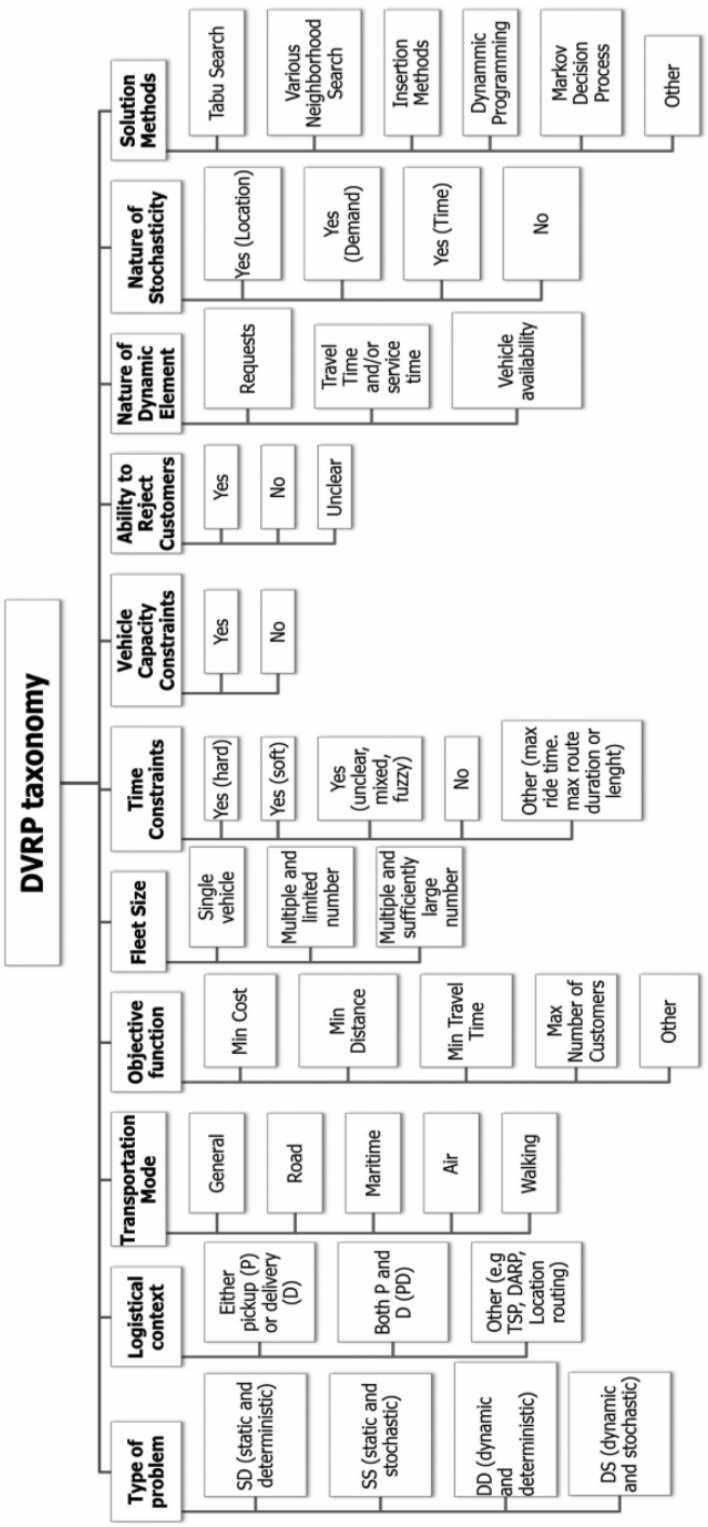

Figure 2.1 – Taxonomy of the DVRP proposed in Psaraftis *et al.* [89]

This section focuses on dynamic methods applied to the PAP. Note that we avoid using the term dynamic PAP in order to remain consistent with the fact that not all papers deal with the PAP as a dynamic problem. Table 2.I is based on the taxonomy depicted in Figure 2.1 and summarizes the solution methods of the PAP formulations detailed in previous sections. Further, we can see that the same criteria for the classification of the DVRP can be used for the PAP, such as the type of problem and time constraints. From the 11 categories of the DVRP listed in [89], we make use of six, see Figure 2.1. We excluded five categories from the DVRP: the logistical context, transportation mode, fleet size, vehicle capacity constraints, nature of stochasticity because their use is less clear in the PAP context.

Paper	Type of problem	Objective function	Time constraints	Ability to reject	Dynamic element	Solution method
Abidi *et al.* [1]	SD	Min cost/distance	Yes (soft)	No	None	Hybrid heuristic
Amer & Chow [2]	None	Social optimum	Unclear	No	Unclear	A106 protocol
Ayala *et al.* [7]	SS	Other	No	Yes	None	Game theory
Geng & Cassandras [39]	DD	Min cost/distance	No	No	Requests	Other (queuing theory)
Mladenović *et al.* [75]	SD	Min distance	Yes (hard)	Yes	None	VNS
Mladenović *et al.* [77]	DD	Other	Yes (hard)	Yes	Requests	Other
Roca-Riu *et al.* [91]	SD	Min distance	Yes (hard)	No	None	Exact
Shao *et al.* [95]	SD	Other	Yes (hard)	Yes	None	Exact
Tang *et al.* [102]	DD	Min cost/distance	Yes (soft)	No	Requests	Other
Teodorović & Lučić [103]	DS	Max revenue	None	Yes	Requests	Fuzzy logic
Verroios *et al.* [110]	DD	Other	Unclear	No	Requests	Heuristics (clustering)
Zou *et al.* [118]	DD	Min distance	Yes (hard)	No	Requests	Design mechanism (game theory)

Table 2.I – PAP literature overview: based on the DVRP taxonomy from Psaraftis *et al.* [89]

The problem type, as defined in [89] relates to the input (static or dynamic) and to the variables of the formulation (deterministic or stochastic). From this point of view four problem types are drawn:

— static and deterministic (SD),

— static and stochastic (SS),

— dynamic and deterministic (DD),

— dynamic and stochastic (DS).

The term *dynamic* is widely used, but in different ways. In [11, 68, 89], as in this thesis, a problem is characterized as dynamic if its input is received and updated concurrently

with the determination of the problem, which in case of [89] is the VRP. The authors further provide an interesting observation, stating: "*given that the definition of the word "problem" in our paper refers to the abstract problem examined in a paper and not to the associated real-world problem, it is conceivable that a VRP may be static whereas its associated real-world problem is dynamic*". This statement, though intuitive, is not a definition of any sort, but a good indication of the void in the literature when a problem is treated as dynamic.

The notion of a objective function in a dynamic setup is also an open topic. In [89] the authors conclude that most papers deal with the objective function of a dynamic formulation in the classical way: "*in a DVRP, one would expect to see a more frequent use of "throughput" or "per unit time" objectives, such as average per unit time serviced customers, average per unit time cost, average demand rejections per unit time, or similar. Yet, and with some exceptions, most of the objectives encountered in the set of reviewed papers are identical or quasi-identical to traditional static objectives*". Note that the objective function column in Table 2.I refers to the objective of the combinatorial formulation and not to the overall (dynamic) objective. This is why in this thesis, and consequently in Table 2.I, we list the solution methods of the dynamic component, if it exists, rather than listing the precise method (exact or approximate) used to solve the decision process (static formulation).

The time constraint category is even less clear though in a dynamic setup the concept of time is essential. But we can say that two different uses of the word "time" can be identified. The first is related to the static formulation, which in itself can contain some time constraints, such as in [91] and the VRPTW. The second time constraints can relate to all the constraints within the dynamic decision making, i.e., handling the decision obtained by solving a (static) formulation at some time moment. We observe that if the formulation is static, i.e., the decision does not change over time, the methods of solving such formulations remain in the heuristic or exact domain. However, if the dynamic element is added, then some authors resort to other methods, such as queuing theory, reverse game theory, Markovian chain process, fuzzy logic, learning techniques or others.

The ability to reject customer is also less strict since a dynamic setup does not need to be a reservation system in which a customer request is accepted or denied. The most

clear-cut example is the revenue management approach of Teodorović & Lučić [103] in which the authors focus on accepting or rejecting vehicle's requests based on the potential revenue. But the main reason to reject a vehicle request to park would be the absence of available parking slots. In this context, it can be assigned to a dummy parking facility if no other feasible allocation can be made, such as in [75, 77]. So, although the request was not denied, it can be considered that there is the ability to reject a customer.

We can conclude that at the heart of the PAP decision process there is a combinatorial problem to solve. However, because of the dynamic nature, there must be another layer which incorporates its dynamic aspects. Table 2.I shows that little attention was payed to include the dynamic element of the PAP, and moreover, that the specific solution methods for dynamic formulations have yet to be fully established in the literature. In this thesis we propose a solution method for the dynamic PAP, which can be generalized to other dynamic formulations.

2.5 New classification of PAP formulations

Previous sections of this chapter allowed us to have a better overview of various dynamic formulations. The PAP is a good example of a dynamic problem because the input parameters, however well estimated, will change over time. This would make any type of static formulation unsuitable in practical terms. In this section we list papers that address the parking allocation of a set of vehicles, whether they are static or dynamic. Moreover, the PAP in any form should contain two (hard) constraints: allocation and capacity. In other words, it must assign a parking spot to all the vehicles while respecting the parking availability. This brings to mind the formulation of the GAP, which also includes these constraints and, as previously discussed, it can be considered as a special case of the VRP. Thus, solution methods applied to the DGAP, and other variants of the DVRP, can be reused for the DPAP.

As mentioned in Pillac *et al.* [86] and Psaraftis *et al.* [89], the number of papers dealing with dynamic formulations has grown dramatically since the beginning of the century. In Section 2.4 we used the taxonomy of Psaraftis *et al.* [89], however, in this section we include new indicators. More precisely, in our classification we focus on the following three classification principles:

1. *Infrastructural requirements*: if the authors require a high infrastructure in terms of network availability and/or a multitude of input parameters such as the time the vehicle will remain parked, the weight drivers would attribute to the cost, etc;
2. *Dynamic aspect*: the mechanism used to deploy input updates and thus the solution updates. Moreover, we investigate whether the authors take a myopic standpoint, and how they evaluate the dynamic solutions;
3. *Static aspect*: the decision process is based on allocating parking to vehicles by means of: MP formulations, reservation systems, game-theoretic formulation, etc.

These indicators remind of some previous formulations of the PAP (e.g. [39, 102, 118]), however unlike them, in this thesis we clearly separate the static and dynamic parts of the PAP. Moreover, we formulate the static part as a combinatorial optimization problem and a specific mechanism on a separate decision level which handles changes over time.

Infrastructural requirements The basis of any problem is founded upon the assumptions researchers make before stating its abstraction (model). In the case of the PAP, we inevitably have a set of vehicles and a set of parking lots to be matched. However, which parameters of these two sets will determine them is up to a specific study or to the researchers themselves. In previous sections, we saw that different requirements, i.e., parameters of the vehicle and parking sets, were chosen when formulating the PAP. We will refer to these parameters as *infrastructural requirements*, because most of them are based on assumptions that some data are available to the decision maker. These data are most often based on some technological assumption, or on the fact that the drivers provide additional preferences to the decision maker or to the assistant. Therefore, when considering infrastructural requirements we will be talking about:

— drivers' preferences: their willingness to cruise, amount of money they are willing to spend, time they are planning on staying parked, etc.;
— technological assumptions: the level of connectivity (network capabilities), the percentage of vehicles which are interconnected, the accuracy of the data, etc.

Obviously, if high technological assumptions are made, then the authors have a higher degree of liberty when formulating a PAP. Moreover, they can produce formulations which are not currently applicable in practice, and may never be. On the other hand,

if several drivers preferences are included, then we often end up with complex formulations of the PAP that are not easily solved. This fact becomes even more important when taking into account the fact that the problem should be solved relatively quickly since new requests continuously arrive. Moreover, even if the drivers enter their preferences, such as the projected time they will remain parked, it does not guarantee the accuracy of this information, further compromising the solution. Therefore, a balance of infrastructural requirements is necessary to formulate an accurate and applicable PAP.

Dynamic aspect The dynamic aspect of the PAP (or most dynamic formulations) relies on the three questions. The first question, i.e., how to update the input, can be treated in several ways, but depends mainly on the infrastructural requirements. Questions two and three have a less clear-cut answer, which will be addressed in more detail in Chapter 4. But we can say that, as in the DVRP, the aim is primarily to avoid poor long-term parking assignments by making "good" short-term ones.

In our classification we regroup questions two and three in one and call it the *dynamic mechanism*. The dynamic mechanism is not to be mistaken for the design mechanism used in [118]. Simply put, the dynamic mechanism can be viewed as the solution method in [89]. However, in [89] the solution methods are the elements of a dynamic formulation which solves the combinatorial formulation, see Figure 2.1. The dynamic mechanism in our classification does not serve to solve the combinatorial optimization (static) problem but rather organises the decisions made at the static level. In a way, placing vehicles in queues and then assigning them parking lots, as in [39] is a dynamic mechanism. Or the sequential solving of static PAPs, like in [75, 77] or [102] can also be considered as a dynamic mechanism.

Static aspect Technological and scientific breakthroughs have made it possible to solve complex combinatorial problems very quickly, without even resorting to specific heuristics, but just by using commercially available solvers. Obviously, to apply most of the solvers, an MP formulation is required and not a combinatorial one. However, most of the PAPs have an MP formulation, which makes it possible to use a solver. These MP formulations can be considered as the *static aspect* of the PAP, in which decisions (assignments) are proposed. In a way, they can be considered as the tool that assigns

parking facilities to vehicles. These decisions are then reevaluated in the dynamic layer of the problem.

The MP model itself can contain a time element, i.e., it can be time-dependent. The parameters and variables that depend on the decision moment, i.e., when the decision making is launched, are not considered time-dependent. Unlike [89] we do not specify whether there are time constraints, but rather investigate whether some of the variables or constraints are time-dependent. As we shall see, if the model includes time, it can be used to adjust, to some extent, the upcoming decisions. More precisely, already at the static level, we can avoid some myopic traps by including time in the static formulation. The solving time of the static formulation is very important if the dynamic mechanism is based on sequentially solving its static counterpart. Therefore, in our classification we also include the solution methods of the static formulation.

Article	**Requirements**		**Dynamic**		**Static**		
	Technological	Preferences	Mechanism	Myopic	Model	Solving method	Time-dependent
Abidi *et al.* [1]	Medium	Low	/	/	AP	Heuristic	No
Amer & Chow [2]	Low	High	A106 model	Yes	/	/	/
Ayala *et al.* [7]	Low	Low	Simulation (sequential)	Yes	Stable marriage (game theory)	Nash equilibrium	No
Geng & Cassandras [39]	Medium	Medium	Simulation (queuing)	Yes	AP	Exact	No
Mladenović *et al.* [75]	Medium	Low	/	/	AP	Exact and heuristic	Yes (other)
Mladenović *et al.* [77]	Medium	Low	Simulation (sequential)	Yes	AP	Exact and heuristic	Yes (other)
Roca-Riu *et al.* [91]	Low	Medium	/	/	VRPTW	Exact	Yes (time windows)
Shao *et al.* [95]	Medium	Medium	/	/	Other	Exact	Yes (hard)
Tang *et al.* [102]	High	High	Simulation (sequential)	Yes	AP	Heuristic	No
Teodorović & Lučić [103]	Medium	Low	Simulation (sequential)	Yes	Other	Exact	Yes (soft)
Verroios *et al.* [110]	High	High	Simulation (other)	Yes	TVTSP	Heuristic	Unclear
Zou *et al.* [118]	Medium	High	Simulation (other)	No	AP	Exact	Yes (hard)

Table 2.II – Classification of existing PAP formulations

In Table 2.II the rows represent different PAP formulations, while the columns represent the seven categories of the previously introduced indicators: requirements, dynamic and static aspects. From it, we observe that there is no common feature of the categories except that most of the formulations are myopic. Even the myopic behavior is not that straightforward since some static models can assign a parking lot to a vehicle in a future time moment and therefore, to some extent anticipate long-term poor decisions. One other observation can be that most authors opted to sequentially solve the static problem,

i.e., stated an online dynamic problem. However, most papers do not take full advantage of this approach and do not reevaluate static decisions of each cycle.

This section has demonstrated that there are many ways of formulating and solving the PAP. All authors agree that the problem is highly dynamic, but as in other dynamic formulations (see Section 2.2) no clear consensus exists on how to formulate or solve it. Most authors formulate a MP model to assign parking facilities to vehicles and then sequentially (periodically) reapply the same model with the updated input. Instead of focusing on this dynamic mechanism researchers mostly focus on techniques for solving the static part of the PAP. Moreover, myopic traps and the quality of the solutions depend on the system's requirement. Some authors impose high requirements, while others are more moderate. We can see that the observation from Spivey & Powell [100] about the balance between completeness and simplicity is crucial.

2.6 Conclusion

Dynamic formulations and corresponding solution methods have been a hot topic in OR and transportation sciences in recent years. We witness an abundance of papers dealing with various dynamic analogs of well established problems in transportation, whose static counterparts have been studied for decades. For example, the first DVRP variant was published in an MIT technical report by Wilson and Colvin in 1977. Nonetheless, there is no standard dynamic formulation for these problems. The same observation can be made for the PAP, which in its core is a variant of the GAP, since two main constraints must be satisfied: allocation and capacity.

This chapter addressed the topic of dynamic formulations, starting from the simplest problem: the AP, until reaching the PDP. We see that there are many ways to tackle these dynamic formulations: sequentially solving a MP model one at the time, predefining an end game design, setting up a Markov chain process, etc. A recurring observation is that at the core of the decision process lies a clear combinatorial formulation which needs to be solved. Then, the dynamic aspect steps in and intervenes once the decision is made. This higher level of decision making can then modify these decisions trying to avoid myopic traps, which would have produced significantly suboptimal solutions in future time instants.

From the classification of the DVRP and following the incremental introduction of dynamic formulations (from the simplest AP to the VRP) we were able to first classify papers dealing with the PAP and then to introduce new indicators specific to the PAP. These indicators take into account the infrastructural requirements which provide us with the data needed to formulate the combinatorial formulation, the static formulation which serves as a tool for allocating parking lots to vehicles, and finally, the dynamic aspect which then evaluates the decisions of the static solution and ensures transitions from one input to the next. The following chapter will first address how we tackled the static part of the PAP. Chapter 4 will further detail on how the dynamic configuration was set for the PAP.

CHAPTER 3

THE STATIC PARKING ALLOCATION PROBLEM

This chapter is the first one completely dedicated to the real-world problem addressed in this thesis, namely the problem of efficiently allocating parking lots to vehicles. We have seen in the previous chapter that the problem of assigning parking lots to vehicles is constantly updated and that it should incorporate a static mathematical programming model that allocates parking to vehicles for a set time instant. Our aim is to propose a formulation that is both theoretically justified and can be practically deployed. To do so, we first introduce a mathematical programming model which assigns parking facilities to vehicles for a given time moment, i.e., a static model. Clearly, the real-world problem is dynamic, but in order to get there, we first analyse the static variant, whereas Chapter 4 deals in detail with the dynamic case. In this chapter, we demonstrate that if we assume that the arrival times of a set of interconnected vehicles is known, then the 0-1 programming model possesses the integrality property. This is in contrast with decentralized networks where the data would be just partially visible. The concept of interconnected vehicles means that the data are visible to a central agency that would have insight to the vehicles whereabouts and is the most common approach in the literature. These assumptions are realistic because most vehicles have some GPS device which can accurately calculate arrival times. The proposed 0-1 programming model contains only the necessary constraints: allocation and capacity. It can then be adjusted if other requirements are imposed, such as preferences or pricing. Moreover, the assumptions we make are realistic, i.e., we do not require some new technology or infrastructure to be put to use. We investigate as well several potential objective functions to evaluate the complexity of the model. Finally, even thought the proposed model possesses the integrality property, if a huge number of parking requests show up, a heuristic proves to be a good alternative to exact methods. We therefore develop a variable neighborhood search heuristic to tackle large instances and perform tests on randomly generated instances to validate the examine the performance.

Chapter outline This chapter introduces the assumptions which will be used in all subsequent chapters and is organized as follows: Section 3.1 introduces the preliminaries of a centralized static problem for connected vehicles; Section 3.2 formulates the 0-1 model; Section 3.3 offers solution techniques for solving it; Section 3.4 of this chapter presents the computational results over a range of randomly generated instances, while Section 3.5 concludes this chapter.

3.1 Preliminaries

Drivers equipped with a GPS device usually enter their final destination into their devices. However, they rarely park their vehicles exactly at this destination point, but more likely at the most convenient available parking slot they can find. The driving time between the desired destination and the actual parking is known to produce several undesirable consequences, such as air pollution, traffic congestion and stress. Detailed urbanization and transportation studies (e.g. [14, 23, 37, 98, 99], see Chapter 1) clearly confirm the negative impact of massive unorganized (random) search for parking in urban areas and advocate the use of parking lots over on-street parking. Therefore, we avoid allocating curb-side parking to vehicles and focus only on parking lots, see Figure 1.1.

We can take advantage of the fact that most 3G/4G/5G devices (e.g. smartphones) have a GPS signal to formulate a 0-1 programming model. This model will be, by design, simple to solve to enable a quick response to the frequent GPS updates. However, if the number of vehicles surpasses a certain threshold, no matter the efficiency of the solver, a heuristic approach is welcomed. That is why in addition to the exact methods we developed several VNS-based heuristics to cope with large scale randomly generated instances, see Section 3.3. Furthermore, it is shown how a complex combinatorial problem, which was usually solved by simulation, can be modeled as a simple 0-1 linear program when the GPS data is included.

We opt for the centralized approach, meaning that the information of a set connected vehicles will be available to a central intelligence (server). Furthermore, we consider allocating parking lots to vehicles at a fixed time point, independently from previous or future decision moments, i.e., we consider the static case of the PAP. However, the static PAP includes vehicles arrival time, and thus is time-dependent, but not yet dynamic.

Practical description of the problem We assume that a set of interconnected vehicles is given with a central dispatcher (server). We have their current positioning on the map (GPS) and their final destinations. These data provide us with their respective traveling times: the driving time to all the parking lots and the walking time from all the parking lots to their destinations. The driving time provides us with the clock time a vehicle

would arrive to a parking lot and thus we would be able to check if there would be an available parking slot at that time. Each parking lot is characterized by its coordinates and capacity. The capacity can be residual, i.e., the capacity for each time of a given planning horizon, and the total capacity of a parking lot. The problem we wish to solve is: with these data, allocate parking lots to vehicles at their (future) arrival time, such that the total traveling time, i.e., driving and walking time, of all the drivers is minimal, respecting the capacity constraints. Since the residual capacity for a given time horizon is provided as input it includes all the vehicles that used those facilities during that period, and not only the ones that depend on our system. This is why flow constraints cannot be imposed, because we do not know which vehicle entered or exited a facility, just the total number of parked vehicles at that time moment. Most of the results presented in this chapter are a collection of our publication [75].

3.2 Problem formulation

We first present a combinatorial formulation of the static PAP, which will be later used for developing a heuristic. We then propose a mathematical programming formulation which is used to solve the problem with some commercial solver, such as CPLEX.

Combinatorial formulation

Assume that n connected vehicles, equipped with a GPS signal, are searching for parking in an urban area at a given time moment. Also assume that there are m parking lots j, each with a known total capacity $q_j, j = 1,...,m$. Once all drivers enter their destinations, we are then able to determine two types of estimated times or distances matrices:

— $T' = (t'_{ij})$: estimated time needed by vehicle i to reach parking j, $i = 1,\ldots,n; j = 1,\ldots,m$;

— $T'' = (t''_{ij})$: estimated walking time from parking j to the final destination of driver $i, i = 1,\ldots,n; j = 1,\ldots,m$.

Additional input is required regarding the estimated number of free slots c_{jt} at parking j, for each time t, $t = 1,...,T_j$, where $T_j = \max_i \{t'_{ij}\}$.

Residual capacity One of the hard constraints that should exist in any PAP model is the capacity constraint. The most obvious capacity constraint is the one which ensures that the total capacity of the parking lot will not be exceeded. But if we suppose that the arrival time of each vehicle is known, then we can also consider a residual capacity, denoted by $C = (c_{jt})s, j = 1,\ldots,m$, for each time of arrival t. This would allow us some more insight into the availability at the vehicles respective times of arrival, instead of verifying the total capacity at some fixed time. By introducing this residual capacity our PAP model is time-dependent, because it allocates parking lots to vehicles taking into consideration the time they are expected to arrive there.

Objective function Let $x(i)$ represent the index of the parking to which vehicle i is allocated, and let $\mathscr{P}$ be a feasible partition of $x = (x(1),\ldots,x(n))$. Our goal is to determine an allocation variable x (or a partition of x into a number of groups less than or equal to m) that minimizes the cumulative traveling time of the vehicles from their initial position to their destination:

$$\min_{x\in\mathscr{P}} f = \sum_{i=1}^{n} (t'_{i,x(i)} + t''_{x(i),i}). \tag{3.1}$$

Feasibility Denote by b_j the number of occupied slots at parking j in the current solution x, and by u_{jt} the remaining number of free slots at parking j at time t, regarding the solution x. The following two properties state feasibility conditions. The first property provides conditions on valid input data which are easy to verify.

Property 1. *A problem instance has no feasible solution if one the following two conditions is met:*

$\sum_{j=1}^{m} q_j < n$

$c_{jt} > q_j, \quad t = 1,\ldots,T_j, j = 1,\ldots,m.$

Proof: There is no feasible solution if the number of vehicles is larger than the number of parking lots. Besides, the capacity q_j of each parking j should not be smaller than the available space for any period t. ◇

The next property gives obvious feasibility conditions which depend on the solution x as well.

Property 2. *The feasibility of partition $\mathscr{P}$ is satisfied if the following two conditions are met:*

- $b_j \leq q_j$ *: the number of vehicles b_j parked at parking j should be less than its capacity q_j, for all j;*
- $u_{jt} \leq c_{jt}$ *: the number of vehicles parked at time t at parking j should be less than or equal to the maximum allowed number c_{jt}.*

Estimating the residual capacity We assume that the c_{jt} values are known and deterministic. In other words, we assume that some statistical investigation has already been performed to determine these values at each minute (or every five minutes) during the day. For example, it is well known that the random variable which represents the time between two consecutive arrivals or departures (of vehicles) to or from the parking is exponentially distributed ($f(t) = \lambda e^{-\lambda t}, t \geq 0$). The parameter λ is estimated by known statistics which use data collected by measuring inter-arrival (or departure) times over several full days. Therefore, knowing the $\lambda_1, \ldots, \lambda_m$ values for each parking lot j and for each time t, allows us to compute the number of free slots c_{jt}. To conclude, the static PAP relies both on the arrival times at the parking and at the final destination, and the number of available slots at each future moment. The final result is an allocation variable x_i: vehicle i should go to parking x_i, and the GPS could guide the driver to its designated parking lot.

Dummy parking lot If there is not enough slots to accommodate all the vehicles, then the model would not have a feasible solution (Properties 1 and 2). An obvious way to avoid infeasible solutions is to introduce a dummy parking lot $j = m+1$. It should have a large capacity, and be far away, i.e., arrival times $t'_{i,m+1}$ are very large for all vehicles i. So whenever vehicle i cannot be parked at any parking lot $j = 1, \ldots, m$, due to the lack of available parking slots, it will be allocated to the dummy lot $j = m+1$. In other words, this penalization of the time of arrival in fact serves to allocate a dummy lot to a vehicle only if no other allocation can be made. Note that the LP model, presented in the following section, incorporates by default the dummy parking lot, providing feasibility for any input. In this way, we avoid infeasible solutions and temporarily place vehicles in the dummy parking lot. Furthermore, the dummy lot can be seen as a buffer for future

allocations. Throughout of this manuscript, if we refer to a solution as infeasible, this means that at least one vehicle was assigned to the dummy lot.

Mathematical programming model

As previously stated, the main purpose of the static PAP is to allocate the best parking j to each vehicle i by minimizing the total traveling time of all the vehicles. The binary decision variable x_{ij} equals to 1 if and only if such an allocation is made. The traveling time of a vehicle represents the sum of the driving time to its parking lot and the walking time from it to the final destination. The objective is to minimize the total traveling time:

$$\text{minimize} \sum_{i=1}^{n} \sum_{j=1}^{m+1} \left[t'_{ij} + t''_{ji}\right] x_{ij} \tag{3.2}$$

subject to

$$\sum_{j=1}^{m+1} x_{ij} = 1, \quad (i = 1, \ldots, n) \tag{3.3}$$

$$\sum_{i=1}^{n} x_{ij} \leq q_j, \quad (j = 1, \ldots, m) \tag{3.4}$$

$$\sum_{i=1}^{n} \alpha_{ij}^{t} \, x_{ij} \leq c_{jt}, \quad (j = 0, \ldots, m, \; t = 1, \ldots, T_j) \tag{3.5}$$

$$x_{ij} \in \{0, 1\}, \qquad (i = 1, \ldots, n, \; j = 1, \ldots, m) \tag{3.6}$$

where

$$\alpha_{ij}^{t} = \begin{cases} 1 & \text{if } t = t'_{ij} \\ 0 & \text{otherwise.} \end{cases}$$

Constraints (3.3) require that every vehicle be parked (allocation constraints), while constraints (3.4) ensure that the number of vehicles allocated to parking j does not exceed parking capacity q_j. Constraints (3.5) guarantee that the vehicle will find an available slot at their arrival time. The 0-1 programming model (3.2)–(3.6) is time-dependent, but is a GAP, instead of the multi-dimensional AP. This is achieved by introducing the binary parameter $\alpha = \alpha(i, j, t)$ which takes the value of 1 if the arrival time of vehicle i

at parking lot j is t. The first paper to include arrival vehicle arrival times in such a way was published in [71].

The dummy parking lot $j = m+1$ has a large capacity, $q_{m+1} = n$, for example with sufficient slots at every future time step $c_{m+1,t} = n, \forall t$, and with larger arrival times $t'_{i,m+1} > M$, for all i and for some M. If a feasible solution exists, then the dummy parking will remain empty. Due to high values of t'_{im+1}, the model will assign a vehicle to $m+1$ if and only if there is no other slot available. Otherwise, some drivers would remain without a parking slot and would be temporarily rejected. Note that only the allocation constraint (3.3) and the objective function (3.2) are affected by the dummy facility $j = m+1$, since the other constraints are always met.

Properties of the static PAP model The static PAP may be presented as a weighted bipartite graph with two types of vertices: vehicle vertices $i = 1,..,n$ and parking vertices $j = 1,..,m$, having weights $w_{ij} = t'_{ij} + t''_{ij}$. We will now prove the property that makes the Boolean model (3.2) – (3.6) easy to solve.

Property 3. *The integer programming relaxation of the Boolean model* (3.2)–(3.6) *has integer solutions* $x_{ij} \in \{0,1\}$, $i = 1,..,n$; $j = 1,..,m$.

Proof: Let A' be the matrix defined by constraints (3.3) and (3.4):

$$A'_{(m+n)\times(mn)} = \begin{bmatrix} 1 & \cdots & 1 & & & & & & & & \\ & & & 1 & \cdots & 1 & & & & & \\ & & & & & & \ddots & & & & \\ & & & & & & & & 1 & \cdots & 1 \\ 1 & \cdots & & 1 & & \cdots & & & 1 & & \\ & \ddots & & & \ddots & & & \cdots & & \ddots & \\ & & 1 & & \cdots & 1 & & \cdots & & & 1 \end{bmatrix}.$$

It is clear that A' is totally unimodular (TU), since all x_{ij}, when summed up over $i = 1,\ldots,n$ and $j = 1,\ldots,m$ are 0 or 1 (with exactly two non-zeros coefficients in each column). Therefore, based on the well-known theorem from integer programming (see e.g. [94]), the problem defined by (3.2)–(3.4) and (3.6) has the integrality property. This means that the Linear Programming (LP) solution (3.2)–(3.4) and (3.6) $0 \leq x_j \leq 1$ is

equivalent to the integer solution of the problem (3.2)–(3.4) and (3.6). In addition, if A' is TU then, $[A'|I]^T$ is also unimodular ([41]). Since the matrix A'' defined by constraints (3.5) may be transformed into an identity matrix by permuting its rows, we conclude that the matrix defined by (3.2)–(3.5) is TU and thus possesses the integrality property. $\diamond$

Possible extensions of the static PAP From an integer programming standpoint, the basic mathematical programming model (3.2)–(3.6) is easy to solve. Moreover, it requires only the basic information from the drivers: their coordinates and destination. As mentioned in Chapter 2, Section 2.5 various degrees of requirements can be imposed. If we require more information from the drivers (drivers preferences from Table 2.II) the static model can be altered to respond more adequately to the users needs. Here we discuss some possible extensions of the basic model (see [74]).

- A time limit for each driver from this allocated parking to its final destination could be introduced, rendering the model even lighter to solve.
- If other transportation options are offered from the parking to the final destination, the problem will become a multimodal transportation problem. For example, drivers could consider taking a bicycle, an EV, or public transportation, as opposed to only walking to their destination.
- Our model is of the min-sum-sum type. Another representation would be the following min-max-sum model: allocate each vehicle to its parking lot to minimize the maximum time a vehicle spends to arrive at its parking:

$$\text{minimize } f(x) = \max_{i=1,..,n} \sum_{j=1}^{m} \left[t'_{ij} + t''_{ij}\right] x_{ij} \tag{3.7}$$

or

$$\text{minimize } f(x,z) = z \tag{3.8}$$

subject to

$$\sum_{j=1}^{m} \left[t'_{ij} + t''_{ij}\right] x_{ij} \leq z \quad (i = 1,\ldots,n), \tag{3.9}$$

and constraints (3)–(6). This would correspond to the bottleneck PAP (see Sec-

tion 2.1). Note that, the min-max-max formulation

$$\text{minimize } g(x) = \max_{i=1,..,n} \max_{j=1,..,m} \left[t'_{ij} + t''_{ij}\right] x_{ij} \tag{3.10}$$

would yield the same solution as the min-max-sum formulation due to constraints (3.3). Indeed, the vehicle that spends the most time to reach its parking (which should be minimized - min-max-max model) is the same as the one identified in the min-max-sum model since all x_{ij} when summed up over j are equal to 0, except for one vehicle. However the min-max-sum model should be considered, since it contains n additional constraints (3.9), and not $n \times m$ as for the min-max-max formulation. Note that the min-max-sum model does not possess the integrality property.

- Some drivers could give more importance to the walking time, i.e., they would prefer to walk less. One way to address this case could be to assign a preference or weight to each driver that would favor walking or driving time, depending on how they value their time and energy. This premise would produce the following objective function:

$$\text{Minimize} \sum_{i=1}^{n} \sum_{j=1}^{m} [(1 - w_i)\, t'_{ij} + w_i\, t''_{ji}] x_{ij}, \tag{3.11}$$

where $w_i \in [0,1]$. Thus lower the value of w_i, the bigger priority is given to walking. The objective function (3.11) keeps the integrality property, but requires drivers to enter their preferences.

- Similarly, the price per parking $j, j = 1, \ldots, m$ can be included into the objective function without deteriorating the integrality property. However, in our tests we have detected that the prices of parking (for less than 24 hours) do not significantly vary from parking to parking in a city, and that the cost of fuel would surpass the cost of parking.

3.3 Solving the static PAP

In this section, we develop a VNS-based heuristic for the PAP. We first discuss why a heuristic approach is useful, despite the fact that the min-sum static PAP model possesses the integrality property (see Property 3). Then we introduce the steps of our VNS-based heuristic, providing detailed pseudo-codes for most procedures.

Heuristic for the PAP

We begin this subsection by showing some similarities between our PAP model and the GAP models found in the literature. We then run CPLEX tests of the GAP with different input to demonstrate that its solving time strongly depends on the input instances.

Generalized assignment problem Our PAP model is a variant of the GAP, which has been proven to be NP-hard, see [93] for a short survey. Therefore, many heuristics have been proposed for solving it, e.g., [17, 67, 83, 93, 114]. There exist several benchmark instances for the GAP, of which one of the most well-known is in the ORlib test instance library Beasley [8]. However, the GAP is highly sensitive to its input parameters. More precisely, in some cases it can be trivial to solve. For example, it is polynomial if it possesses the integrality property, as in our PAP model. We briefly examine how different GAP instances from the literature behave when solved by CPLEX. We found that all instances proposed in ORlib could be solved at the root of branch-and-bound (B&B) tree. The reason could be either the integrality property or the existence of an efficient heuristic within CPLEX which closes the integrality gap at the root of a branch-and-bound tree, see the last two lines of Table 3.I.

n	m	Instance	Type	Execution time	B&B nodes
1600	80	Type E	(Min)	579.14	5823
1600	20	Type E	(Min)	13.54	1607
900	15	Type D	(Min)	602.23	345679
400	20	Type D	(Min)	602.12	489395
60	10	ORLIB	(Max)	(aver) 0.37	(aver) 0
50	10	ORLIB	(Max)	(aver) 0.27	(aver) 0

Table 3.I – CPLEX results for different types of GAP instances

This is why new set of instances, called *A, ..., E type* were proposed in [114]. The authors showed that GAP can be challenging even for relatively small instances, see Table 3.I. This means that if additional constraints were added to our model, as in the min-max models, or if the model has floating point input, it will be much harder to solve, and thus a heuristic approach becomes a preferable option. Furthermore, the size of the instances alone can present a serious obstacle for a solver (see [68]). In other words, even if the model itself can be quickly solved by a solver, it takes more time for the solver to load the input and create the model, than to effectively solve it.

Another strong argument for developing a heuristic for the PAP is the fact that in large cities there could be more than 100,000 vehicles on the streets looking for a parking place. The model could have millions of variables and just transferring the data to the central server would be excessively time consuming. In such cases, even a greedy heuristic followed by a local search heuristic could provide satisfactory solutions.

Variable neighborhood search

Before presenting our implementation of VNS for the PAP, we present a brief methodology of VNS and survey some papers where VNS was used to successfully implemented on some well-known problems. A survey paper on the VNS 0-1 MIP heuristic framework can be found in [45].

For a given optimization problem

$$\min\{f(x) \ : \ x \in X,\, X \subseteq S\}, \tag{3.12}$$

S, X, x and f denote respectively the solution space, feasible set, a feasible solution and a real-valued objective function. If S is a finite set, (3.12) is a combinatorial optimization, while if $S = R^n$, then (3.12) is a continuous optimization formulation. In this manuscript we focus only on discrete formulation, i.e., where $|S| < \aleph_0$. A solution $x^* \in X$ is optimal if

$$f(x^*) \leq f(x), \quad \forall x \in X.$$

For a given neighborhood $N \subseteq X$, the solution $x' \in X$ is a local optima if

$$f(x') \leq f(x), \quad \forall x \in N(x).$$

The local search (LS) heuristic consists of finding a direction of descent from that solution within a given neighborhood $N(x)$ and moving towards the minimum of $f(x)$ within $N(x)$. Usually the steepest direction of descent (best improvement) is used in which case the local optimal is reached. The steepest direction requires iterative moves towards the local optima until their is no more a direction of descent. In some cases, there can be a stopping criteria for the LS, before reaching the optimal solution for the neighborhood N. Such an alternative direction can be the first descent (first improvement), but other conditions can also be considered such as the number of iterations without improvement, or the execution time. Observe that a neighborhood structure $N(x)$ is defined for all $x \in X$. In discrete optimization problems it usually consists of all vectors obtained from x by some simple modification, e.g., in the case of 0–1 optimization, complementing one or two components of a Boolean vector.

Variable neighborhood search is a framework, i.e., metaheuristic for solving continuous and discrete optimization problems, proposed in [78, 79]. Its methodology is based on three observations ([46]):

(i) a local minimum for one neighbourhood structure does not have to be a local minimum for another neighbourhood structure;

(ii) the global minimum is a local minimum with respect to all possible neighbourhood structures;

(iii) for many problems the local minima of several neighbourhoods are relatively close to each other.

These premises motivate the change of neighborhood in the search for the global optima of a given optimization problem. This principle of changing neighborhoods is used in both the diversification and intensification phase of the VNS metaheuristic. These simple principles led to many variations of VNS. The basic variant ([46, 47]), also referred to as the basic VNS is illustrated in the following algorithm:

Algorithm 1 VNS(x, k_{max}, t_{max})

```
1:  procedure VNS
2:      repeat
3:          k ← 1;
4:          repeat
5:              x' ← Shake(x,k);                                  ▷ Diversification
6:              x'' ← LS(x');          ▷ Intensification (first or steepest descent direction)
7:              x ← NeighbourhoodChange(x,x'',k);                 ▷ Change neighbourhoods
8:          until k = k_max
9:          t ← Execution Time;
10:     until t > t_max
11: end procedure
```

In Algorithm 1 x denotes the initial solution upon which VNS will iterate until all k_{max} neighborhoods are visited. The `NeighbourhoodChange` procedure keeps the neighbourhood k if x'' represents an improvement, otherwise it changes neighbourhoods to $k+1$. For neighborhood structures $N_k, k = 1, \ldots, k_{max}$, the only parameter left to add is the maximal execution time, denoted by t_{max}. Therefore, for given neighborhoods N_k and their respective local searche the only two parameters of the basic VNS are t_{max} and k_{max}. These simple principles led to the successful applications of the VNS metaheuristic in various problems, such as the unit commitment, location problems, routing problems, etc. ([29, 30, 44, 70, 105–107]).

PAP solution representation We present our solution of the static PAP as an array, that was already defined in the combinatorial formulation section:

$x = (x_1, \ldots, x_n)$, where x_i defines the parking lot to which vehicle i is allocated ($x_i \in \{1, \ldots, m\}$).

In order to efficiently compute (update) objective function values associated to solutions in the neighborhood of x and to check their feasibility, we keep, along with the solution x, the following variables:

— f_{cur} : the objective function value of the current solution x;

— $f_v(i)$: contribution of vehicle i to the objective function value ($f_v(i) = t'_{i,x(i)} + t"_{i,x(i)}$);

— $b(j)$: the number of used parking slots at parking j in the current solution x;

— $u(j,t)$: the number of free parking slots at parking j at time t in solution x.

Initial solution In order to construct an initial feasible solution we propose a *Greedy add* algorithm. For each vehicle i we find its closest parking $o(i,1)$; if not feasible (i.e., the parking is full at arrival time $t'_{io(i,1)}$), the vehicle is allocated to its second closest $o(i,2)$, etc. Its steps are presented in Algorithm 2.

Algorithm 2 Greedy Add(T',T'',C)

```
 1: procedure GREEDY_ADD
 2:     u ← v  (u(j,t) ← v(j,t), ∀j,t)                          ▷ Initialize auxiliary variable u
 3:     Get order matrix O_{n×m} = o(i,j)                       ▷ Sort by total traveling time (t' + t'') each row
 4:     Get order p(i) of vehicles o(i,1), (∀i)                 ▷ Sort by column, i.e, in ascending value per vehicle
 5:     b(j) ← 0,∀j; f_v(i) ← 0,∀i; f_cur ← 0; tt ← 0           ▷ Initialize auxiliary variables
 6:     for each vehicle ii = 1  to  n do
 7:         i ← p(ii);                                          ▷ Get vehicle index in ascending order
 8:         tt ← tt + 1;                                        ▷ Update index of parking
 9:         if (tt > m) then 'No feasible solution' stop        ▷ Check if all parking lots were checked
10:         x(i) ← o(i,tt); j ← x(i); t ← t'(i,j)               ▷ Place vehicle i on parking tt and update variables
11:         if (u(j,t) = 0  or  b(j) + 1 > q_j) goto 8          ▷ If capacity if full, test next parking lot
12:         f_v(i) ← t + t"(i,j); f_cur ← f_cur + f_v(i);       ▷ Update variables
13:         u(j,t) ← u(j,t) − 1; b(j) ← b(j) + 1;  tt;← 0       ▷ Update variables
14:     end for
15: end procedure
```

In line 3 of Algorithm 2, for each vehicle i, the parking places are ranked in non-increasing order of their distances from the vehicles. This defines the matrix O, where the element $o(i,1)$ represents the index of the parking lot closest to vehicle i, $o(i,2)$ is its second closest, etc. In line 4 we rank the vehicles based on the distance to their closest parking. This permutation of the set of vehicles is denoted by $p(i)$. In line 5 we initialize arrays b, f_v and f_{cur}. The allocation of each vehicle starts from line 6, following the order obtained by the permutation p. The feasibility is checked in line 9: there should be an available slot at parking j at time t. If it is not feasible, we try to allocate to the next closest parking of vehicle i. If the allocation is feasible, we update the solution, as presented in lines 12 and 13.

Property 4. *The complexity of the Greedy add algorithm is* $O(nm\log m)$.

Proof: For each of the n vehicles, the order of all m parking lots is found in line 3. Hence, its complexity is $O(nm\log m)$, since ordering of array with m elements is in

$O(m \log m)$. The complexity of line 4 is then $O(n \log n)$. The complexity of the allocation loop from line 6 to 14 is in $O(nm)$ since in the worst case the vehicles will be allocated to their furthest parking. Thus, the most time consuming operations are performed in line 3. ◇

As mentioned earlier, we introduce a dummy parking lot to avoid generation of infeasible solutions. Basically, the model structure does not change. However, after introducing a dummy variable, the code would never stop in line 9 of GREEDY_ADD procedure, since feasibility in line 12 is always ensured by the dummy variable, if not before. Moreover, another interesting property may be observed.

Property 5. *The number of vehicles allocated to the dummy parking obtained by Algorithm 2 is the same in the optimal solution.*

Proof. Let us denote by $\alpha(Greedy)$ and $\alpha(Exact)$ the number of vehicles parked after the Greedy and the Exact procedures, respectively. Due to the large values of $t'_{i,m+1}, \forall i$, we have $\alpha(Greedy) \geq \alpha(Exact)$. Suppose the opposite from the claim of this property, i.e., assume that $\alpha(Greedy) > \alpha(Exact)$. This means that there should be free parking slots derived by Greedy solution equal to the difference $k = \alpha(Greedy) - \alpha(Exact) > 0$. Denote with i such a vehicle. The inner loop defined by lines from 8 to 11 of GREEDY_ADD excludes the possibility that i can be moved out from the dummy parking lot. Indeed, for such a vehicle i, variable $tt = \alpha(Greedy)$ in the pseudo-code increases until it reaches m (there is no parking slot j in time moment t for vehicle i). Therefore, $k = 0$, which is a contradiction. ◇

This interesting property tells us that if the greedy solution includes vehicles allocated to the dummy parking lot, then its number cannot be reduced by trying to get a better solution. The better solution could possibly be obtained by allocating different vehicles to the dummy parking lot. So, if the objective is to minimize the number of vehicles without a parking slot, the greedy solution is optimal. This fact is another argument for using a heuristic approach in solving a relatively simple static PAP. An exact solution will not reduce the number of unassigned drivers.

Neighborhood structures Obviously, there can be several neighborhood structures for this combinatorial optimization problem. We propose two neighborhoods:

Allocation: given a solution x and therefore (i, x_i) connections, for each vehicle i, change its parking lot x_i. The neighborhood $N_k^{all}(x)$, can be defined as repeating the reallocation move k times. Therefore, the distance between two solutions x and y is equal to k if and only if they differ in k allocations: $x_i \neq y_i$ exactly for k vehicles; for the remaining $n-k$ vehicles $x_i = y_i$, holds.

Interchange: given a solution x, let (i_1, j_1) and (i_2, j_2) denote two vehicles parking pairs. Assume that vehicles i_1 and i_2 exchange their parking places, so that we have the pairs (i_1, j_2) and (i_2, j_1) in the new solution y. The 1-interchange neighborhood $N_1^{int}(x)$ consists of all solutions y obtained from x after performing such interchanges. It is clear that not all solutions are feasible since some vehicle could arrive when all parking slots are occupied. We define the k^{th} neighborhood of x, $N_k^{int}(x)$, with respect to the interchange structure as the solutions obtained by k interchanges.

These neighborhood structures for the PAP were first proposed in Mladenović *et al.* [72].

Shaking The shaking step in basic VNS consists of a random move from the current solution x to a solution $x' \in N_k(x)$. We use both neighborhood structures, Allocation and Interchange for the shaking step, with the same probability. In addition, we implement the so-called *intensified shaking* for Allocation neighborhood $N_k^{all}(x)$, where the vehicle is first chosen at random and then its best identified reallocation. This step is repeated k times to reach solution x' from $N_k^{all}(x)$. The complexity of this procedure is obviously $O(m)$.

Allocation local search We perform a LS using a reallocation neighborhood structure. Given a feasible solution x, every vehicle tries to change its parking to every other parking. It is clear that the cardinality of $N_1^{all}(x)$ is $n \times m$. However, we can significantly reduce it in the following way: reallocate vehicles just to r_v (a parameter $r_v < m$) closest parking lots.

In the reduction strategy, used during the preprocessing, we need to rank distances (or times) $t'_{ij} + t"_{ij}$ in non-decreasing order of their values, for each vehicle i and each parking j.Thus obtaining the order of parking facilities $o(i, j)$, $j = 2, \ldots, m$, for each vehicle i. Note that the matrix O has already been introduced for the greedy Algorithm

2. A detailed description of our local search is provided in Algorithm 3.

Algorithm 3 Reallocate LS($x, f_{cur}, f_v, b, o, r, u, first$)

```
1:  procedure REALLOCATE_LS
2:      improve ← true
3:      while improve do                          ▷ While improvements are found continue the LS
4:          improve ← false
5:          for each vehicle i = 1 to n do
6:              jj ← x(i);  tt ← t'(i, jj);  f_new ← f_cur − f_v(i);          ▷ Update variables
7:              for each parking j = o(i,1) to o(i,r) do
8:                  t ← t'(i, j);                                  ▷ Update the arrival time
9:                  if (u(j,t) > 0 & b(j) + 1 ≤ q_j) then          ▷ Check capacity constraints
10:                     f_new ← f_new + t + t"(i, j);              ▷ Update objective function
11:                     if f_new < f_cur then ▷ Check the new objective function improved on the last value
12:                         f_cur ← f_new;  improve ← true;        ▷ Update all variables
13:                         x(i) ← j; f_v(i) ← t'(i, j) + t''(i, j);
14:                         b(j) ← b(j) + 1; u(j,t) ← u(j,t) − 1l
15:                         b(jj) ← b(jj) − 1; u(jj,tt) ← u(jj,tt) + 1l
16:                         if (first) return
17:                     end if
18:                 end if
19:             end for
20:         end for
21:     end while
22: end procedure
```

The input variables in `Reallocate_LS`, among those already introduced earlier in `Greedy_Add` are

— $first$: a Boolean variable which defines whether the first or the best improvement strategy is implemented in the LS;

— r_v : an integer value that defines how many parking we will try to change with the current one, for any vehicle, following their distance order.

The basic loop starts at line 3. It is repeated until no improvement can be obtained in the reallocation neighborhood $N_1^{all}(x)$. For each vehicle i, its current parking jj (at time tt) is replaced with the parking j (at time t). The feasibility of this reallocation is checked in line 9; whether a better solution is found or not is checked in line 11. If the move is not feasible, or if there is no improvement, vehicle i remains at the same parking lot. Otherwise the solution x is updated, together with arrays f_v, b_j and matrix U. If the first improvement strategy is implemented, the procedure returns the improved values in line 16.

The number of iterations of reallocation LS is not known in advance and thus we

do not know the worst-case complexity of this algorithm. However, we can find the complexity of one iteration of Algorithm 3. The following property is obvious:

Property 6. *The number of calculations of one* `Reallocate_LS` *iteration is bounded by* $O(rn)$.

Interchange local search This local search uses $N_1^{int}(x)$ neighborhood described earlier. Detailed pseudo-code is given at Algorithm 4.

Algorithm 4 Interchange LS(x, T', T'')

```
1:  procedure INTERCHANGE_LS
2:      improve ← true
3:      while (improve) do                          ▷ Loop until there is no improvement
4:          improve ← false
5:          for each vehicle i1 = 1 to n − 1 do
6:              j1 ← x(i1); t1 ← t'(i1, j1);        ▷ Update arrival time and allocated parking lot
7:              for each vehicle i2 = i1 + 1 to n do
8:                  j2 ← x(i2);                     ▷ Assign new parking lot
9:                  if j1 ≠ j2 then                 ▷ Check if vehicles are parked at the same lot
10:                     t2 ← t'(i2, j2); t3 ← t'(i1, j2); t4 ← t'(i2, j1);   ▷ Update arrival times
11:                     if (u(j2,t3) > 0 & u(j1,t4) > 0) then   ▷ Check both lots availability
12:                         fnew ← fcur − fv(i2) − fv(i1);   ▷ Update variables
13:                         fv1 ← t'(i1, j2) + t''(i1, j2);  fv2 ← t'(i2, j1) + t''(i2, j1)
14:                         fnew ← fnew + fv1 + fv2
15:                         if fnew < fcur then     ▷ If improvement is found update all values
16:                             fcur ← fnew;  improve ← true
17:                             u(j1,t1) ← u(j1,t1) + 1;  u(j2,t2) ← u(j2,t2) + 1
18:                             u(j1,t4) ← u(j1,t4) − 1;  u(j2,t3) ← u(j2,t3) − 1
19:                             x(i1) ← x(i2); x(i2) ← j1
20:                             fv(j1) ← fv1; fv(i2) ← fv2
21:                             if first return
22:                         end if
23:                     end if
24:                 end if
25:             end for
26:         end for
27:     end while
28: end procedure
```

Note that in the interchange LS consits of two vehicles (i_1 and i_2), two corresponding parking lots (j_1 and j_2), and four different times:

t_1 : the time at which vehicle i_1 arrives at its current parking j_1;

t_2 : the time at which vehicle i_2 arrives at its parking j_2;

t_3 : the time at which vehicle i_1 arrives at parking j_2, and

t_4 : the time moment at which vehicle i_2 arrives at parking j_1.

We need to interchange the vehicle-parking pair (i_1, j_1) with (i_2, j_2) to obtain the (i_1, j_2) and (i_2, j_1) allocations for each feasible pair of vehicles i_1 and i_2. This move is not possible if both vehicles are already at the same parking in solution x (condition $j_1 \neq j_2$ at line 9). Note that we do not need to include the capacity constraints 3.4 here, since vehicles just exchange their parking lots. However, it may occur that at time t_3 or t_4 there will be no slots available. This condition is verified in line 11. The new solution is calculated in lines 12, 13, and 14, and if improved, it is updated in lines 16–20.

In terms of Algorithm 4 complexity the following property is obvious:

Property 7. *The number of calculations in one iteration of* `Interchange_LS` *is bounded by* $O(n^2)$.

Despite the theoretically large number of operations, the algorithm can be very fast due to the fact that many moves are not feasible, and that vehicles from the same parking do not interchange. Moreover, we have implemented the first improvement strategy, further reducing the search time.

Sequential variable neighborhood descent Variable neighborhood descent (VND) is a deterministic variant of VNS. In its sequential version, neighborhoods are placed in a list and used sequentially in the search. The basic VND (BVND) returns the search back to the first neighborhood, whenever an improvement has been detected in any neighborhood structure from the list. The VND method is obtained if a change of neighborhoods is performed in a deterministic way. Most local search heuristics in their descent phase use very few neighbourhoods. The final solution should be a local minimum with respect to all k_{max} neighbourhoods. Hence, the chances to reach the global one are larger when using VND than with a single neighbourhood structure. For the static PAP, our list contains two neighborhood structures in the following order: reallocation and interchange. The BVND is implemented, since Interchange LS uses the first improvement strategy. In other words, the first time interchanging parking lots between two vehicles is successful, the search resumes with reallocation. As in any other deterministic local search, VND stops when the solution is local minimum with respect to both neighborhood structures.

General variable neighborhood search We also implemented VNS, in which the BVND heuristic is used as a local search, i.e., intensification phase. This VNS variant is known as general VNS (GVNS). The basic loop contains the following three steps: Shaking, VND local search and neighborhood change. More precisely, line 6 of the BVNS algorithm 1 can be changed to

$$x'' \leftarrow VND(x, k_{max}).$$

By using different neighborhoods, i.e., BVND in the intensification phase we increase the chances of obtaining better results.

Reduced VNS The reduced VNS (RVNS) heuristic is based on a random choice of points from $N_k(x)$ and no descent is made, i.e., there is no intensification phase. The values of these newly chosen points are just compared with the incumbent one and an update takes place if an improvement occurred. The stopping condition is maximal execution time t_{max} or the maximum number of iterations between two improvements. As BVNS and GVNS RVNS uses only two parameters: t_{max} and k_{max}. RVNS is useful for very large instances, in which the LS can take up much of the execution time.

3.4 Computational results

In this section we perform computational experiments to validate the static PAP model and its properties. To do so, we first generate random instances, which are in turn divided into feasible and infeasible. The latter being those which have at least one vehicle allocated to the dummy lot. Moreover, usefulness of the heuristic becomes obvious on large-scale instances. We therefore examine the static PAP on smaller instances and larger ones, with both feasible and infeasible solutions.

The previously described heuristics were coded in Visual Studio 2012 C++. All tests were executed on Intel Core i7-4702MQ processor with 16GB RAM running on Windows 7 professional platform. CPLEX 12.6 was evoked via concert technology, coded in C++ on Visual Studio 2012 and ran in parallel on all cores, while the heuristics were sequential.

Random test instances

We have tested our model and the VNS-based heuristics on randomly generated test instances. The number of vehicles n varies from 1,000 to 90,000, while the number of parking lots m is $10, 20$, 30 and 50. The maximum capacity Q of each parking is equal to $[2n/m]$. Then, the actual capacity q_j is generated at random between 1 and Q, for each parking j. The vehicles positions and their destinations are generated according to a discrete uniform distribution in the square $S = [0,200] \times [0,200] \in \mathbb{R}^2$. The parking locations are also chosen at random within the same area S. Rectangular distances between all drivers locations to all parking locations are used to generate the $T' = (t'_{ij})$ distances. The distances between parking and destinations $T'' = t"_{ij}$ are computed in the same way. The values of matrix $C = (c_{jt})$ are generated in the following way. The initial values for each parking j at time t_1 are generated from a discrete uniform distribution $v_{jt_1} \in [1, q_j]$. In order to generate more realistic instances, we generate the values $v_{j,t+1}$ using the values c_{jt} for $t = 1,..,T$ (where $T = \max_{i=1,..,n; j=1,..,m} \{t'_{ij}\}$):

$$c_{j\,t+1} = c_{jt} + \gamma,\ \gamma \in [-3,3].$$

In other words, we do not allow the change in the number of free parking slots to be greater than 3, for all parking lots j.

Computational results are divided into two parts. We first compare the exact solutions with the heuristic on small and medium size instances ($n = 1000, 3000, 5000, 7000$ and 9000), for cases where dummy lots are not needed (Table 3.II) and where the input does not produce feasible solutions (Table 3.III). We then switch to larger scale instances, where the number of vehicles searching for a parking lot ranges from 10,000 to 90,000. All the datasets used in these test are available on https://goo.gl/H3Nu5H.

Feasible small and medium size instances

The feasibility of the instances is checked according to Properties 1 and 2. If an instance is not feasible, a new one is generated. In addition, if the greedy algorithm cannot find a feasible solution, we generate a new random instance as well. Thus, all the following instances have feasible solutions.

For the number of vehicles we evaluate five possibilities: n = 1000, 3000, 5000, 7000 and 9000. As mentioned previously, for each value of n, we consider three possible cases of parking: m = 10, 20 and 30. In addition, for the same (n,m) values, we generate 10 instances. Therefore, in total we generate $5 \times 3 \times 10 = 150$ test instances.

Comparison We compare the results in solving static min-sum PAP of the following methods:

— `CPLEX` : exact method using CPLEX solver on model (2)–(6);

— `Greedy` : greedy heuristic described in Algorithm 2;

— `SeqVND` : sequential VND-based local search, as given in Section 3;

— `GVNS` : general VNS, running maximally 10 additional seconds.

Average results on 10 instances, for different pairs of n and m are presented in Table 3.II.

Parameters		Exact	% Error			Running time (seconds)		
n	m	CPLEX	Greedy	seqVND	GVNS	CPLEX	SeqVND	GVNS
1000	10	158203	4.09	0.10	0.08	0.90	0.33	4.71
	20	147250	4.43	0.16	0.14	1.21	0.55	6.43
	30	144064	4.66	0.22	0.19	1.55	0.60	7.40
3000	10	507136	6.28	0.08	0.08	1.98	4.41	3.13
	20	451402	4.29	0.14	0.13	2.88	6.15	2.92
	30	432211	4.95	0.15	0.14	4.28	8.97	4.26
5000	10	822996	5.78	0.07	0.06	2.64	19.12	1.58
	20	728954	3.22	0.10	0.10	4.99	29.76	2.62
	30	729491	5.51	0.12	0.12	7.76	54.06	1.73
7000	10	1131207	4.94	0.22	0.22	3.62	45.75	0.85
	20	1024958	3.81	0.13	0.13	6.54	82.28	2.27
	30	1005248	4.01	0.12	0.12	8.23	133.69	0.00
9000	10	1453969	5.86	0.05	0.05	4.61	75.96	1.24
	20	1329617	4.75	0.09	0.09	9.20	120.09	0.85
	30	1286264	3.92	0.12	0.12	10.90	161.47	0.00

Table 3.II – Average results on ten instances for each n and m.

The third column of Table 3.II provides the optimal solutions of the problem. The next three columns report the percentage deviation from the optimal solution values obtained by `Greedy`, `SeqVND` and `GVNS`, respectively. The next four columns show the

corresponding running times of compared methods. Note that Greedy and SeqVND stop naturally since they are deterministic procedures and that GVNS starts once a solution is provided by SeqVND. Therefore, the total time GVNS spends is the sum of SeqVND and the time provided in the GVNS column. Also note that only ten additional seconds are allowed for GVNS.

The following conclusions may be drawn from Table 3.II. The best method is obviously the exact algorithm CPLEX. This is expected, since we intentionally designed the basic static PAP model to be fast and "integer friendly". The results obtained by SeqVND local search, initialized by Algorithm 2, are very close to the optimal ones (never larger than 0.22%), but for larger sizes this heuristic takes more time than CPLEX. It seems that GVNS cannot easily escape from the deep local minima provided by SeqVND. In more than 50% of the cases it was not able to improve the solution within ten seconds. The solutions provided by Greedy are obtained very fast, i.e., it never takes it more than 0.1 second. The solution quality of this algorithm depends heavily on the instance. If there are a lot of parking slots, which never occurs in our test instances, the solution provided by the greedy algorithm is optimal.

Infeasible small and medium size instances

We now consider instances of the same size as in the previous subsection, but allowing infeasible solutions. The vehicle number n does not exceed the total capacity of all the parking lots ($n \leq \sum_{j=1}^{m} q_j$), but may produce an infeasible input due to current availability c per time step t. Tests are conducted on four instances for each n and $m = 50$. The running time of the RVNS is fixed to five seconds, since in a dynamic version, the time between two runs of the static code should not be large or unpredictable. Note that RVNS does not use any local search. The neighborhood structure used for the perturbation or shaking phase is *Swap*, since *Reallocation* move has no sense in cases where there are more vehicles than parking slots (see Property 5).

We devote this section to infeasible instances mainly because we would like to have an idea of how the heuristics results deviate from the optimal solution, since the number of parked vehicles will remain the same. This is why in addition to the objective function, the Tables 3.III and 3.III include the number of unparked vehicles. Therefore, the instances were specifically designed to answer this scenario.

The results are reported in Table 3.III. Its second column represents the number of vehicles without a parking slot, i.e., the number of vehicles that are parked at the dummy parking. Note that, due to the Property 5, this number is equal for all tested methods. The next three columns report the objective values obtained by `CPLEX`, `Greedy` and `RVNS`, respectively. Columns from six to eight give corresponding computing times spent by the three methods. The last two columns, as in the previous table, provide the percentage of error for two heuristics as $(f_{heur} - f_{exact})/f_{exact} \times 100$.

	# of un-	Objective values			Running time (sec)			% error	
n	parked	CPLEX	Greedy	RVNS	CPLEX	Greedy	RVNS	Greedy	RVNS
1000	3	706592	733584	714842	0.58	0.02	5.00	3.82	1.17
	46	879492	964844	894230	0.56	0.00	5.00	9.70	1.68
	15	743327	800079	759107	0.52	0.00	5.00	7.63	2.12
	18	750107	832561	760679	0.61	0.00	5.00	10.99	1.41
Average	20.50	769879.5	832767.0	782214.5	0.57	0.00	5.00	8.04	1.59
3000	254	2876571	3138545	2900863	2.19	0.02	5.00	9.11	0.84
	275	2969645	3299489	2999241	2.40	0.02	5.00	11.11	1.00
	64	2230716	2343062	2253236	1.98	0.02	5.00	5.04	1.01
	176	2592602	2780696	2614336	1.97	0.02	5.00	7.26	0.84
Average	192.25	2667383.5	2890448.0	2691919.0	2.13	0.02	5.00	8.13	0.92
5000	497	5058142	5434074	5142612	4.74	0.00	5.00	7.43	1.67
	564	5312057	5713841	5410361	4.87	0.00	5.00	7.56	1.85
	63	3629276	3884942	3712448	4.35	0.00	5.00	7.04	2.29
	582	5357472	5862696	5493064	4.61	0.02	5.00	9.43	2.53
Average	426.50	4839237.0	5223888.0	4939621.0	4.64	0.00	5.00	7.87	2.09
7000	757	7279344	7885014	7505630	7.78	0.02	5.00	8.32	3.11
	559	6592231	7149541	6827215	7.05	0.02	5.00	8.45	3.56
	1003	8058449	8794531	8329667	8.35	0.02	5.00	9.13	3.37
	773	7518119	8160985	7793777	7.81	0.02	5.00	8.55	3.67
Average	773.00	7362036.0	7997518.0	7614072.0	7.75	0.02	5.00	8.61	3.43
9000	1549	11283644	12046312	11627640	10.32	0.03	5.00	6.76	3.05
	604	7994571	8535257	8237163	8.43	0.03	5.00	6.76	3.03
	1164	9832471	10622843	10136153	8.84	0.03	5.00	8.04	3.09
	294	7346493	8177867	7854405	14.50	0.03	5.00	11.32	6.91
Average	902.75	9114295.0	9845570.0	9463840.0	10.52	0.03	5.00	8.22	4.02

Table 3.III – Comparison of Exact, Greedy and RVNS methods on small and medium size instances with m=50 parking lots, dummy parking and different number of vehicles

Comparing the results with and without the dummy facility, one can conclude the following: 1) there is no significant difference in effort for obtaining the exact solution

for both sets of instances; 2) as expected, RVNS performs better than `Greedy` for small n. For larger problems, there is not enough time to reach higher precision.

Infeasible large instances

We also compared exact and heuristic methods on instances with $n = 10000$, 30000, 50000, 70000 and 90000, and for $m = 50$ parking lots. Again four instances are generated for each n and $m = 50$. The locations of vehicles and parking lots are taken from the square $[1000 \times 1000]$ and the location of dummy facility is set at the point with coordinates $(1700, 1700)$. Since the solution should be obtained within less than five seconds, among several VNS variants, we run only Reduced VNS after the Greedy initial solution.

It appears that the time it takes to achieve the exact solution on large instances is larger than the operator (dispatcher) can wait. For the number of vehicles ranging from 10 to 50 thousand, despite the polynomial complexity of min-sum-sum PAP, the time needed is in between 10 and 250 seconds. Moreover, for more than 70 thousand vehicles, our PC ran out of memory (16GB). These results confirm the necessity of a heuristic approach for solving real-life problems, even though the problem is not NP-hard. In addition, min-max-max and mix-max-sum are not polynomial problems, and heuristic approach would be even more desirable.

From the previous subsections (Tables 3.II -3.IV) we can conclude proposed heuristics reach a deep local minima, from which they cannot easily escape within the given time limit. Moreover, the gaps do not decrease with the rise of the instance size. This provides further motivation to develop a polynomial algorithm for the PAP. Note that, the basic Hungarian (and its analogs) do not provide optimal solutions for the PAP.

3.5 Conclusion

Searching for available parking lots emerges as one of the major problems in urban areas. The massive unorganized pursuit of parking spaces causes traffic congestion, financial losses, negative environmental effects, among others. Most studies on this topic base their research on simulations, due to mostly non-deterministic input. In this chapter, we propose a new mathematical programming model that makes use of arrival times to parking and destinations as input. These data can be collected by GPS devices of a set of vehicles as input. We call it the static parking allocation problem. We show that

n	# of unparked	Objective values			Running time (sec)			% error	
		CPLEX	Greedy	RVNS	CPLEX	Greedy	RVNS	Greedy	RVNS
10000	1104	10338915	11212253	10749531	10.67	0.03	5.00	8.45	3.97
	796	9359846	10220204	9887008	13.00	0.02	5.00	9.19	5.63
	313	7868790	8327986	8176778	12.16	0.03	5.00	5.84	3.91
	1027	10162548	11060328	10695966	12.48	0.03	5.00	8.83	5.25
Average	810.00	9432524.0	10205192.0	9877321.0	12.08	0.03	5.00	8.08	4.69
30000	1642	25688032	27225734	27139640	61.82	0.08	5.00	5.99	5.65
	3	20484031	20623065	20619435	29.74	0.06	5.00	0.68	0.66
	1521	25558158	27832720	27731256	78.90	0.08	5.00	8.90	8.50
	1425	24874247	26329169	26267431	53.96	0.08	5.00	5.85	5.60
Average	1147.75	24151118.0	25502672.0	25439442.0	56.10	0.08	5.00	5.35	5.10
50000	7569	60307988	66659792	66421052	252.50	0.14	5.00	10.53	10.14
	1362	38926503	41748795	41733403	179.12	0.11	5.00	7.25	7.21
	30	34697447	36491253	36470753	152.78	0.11	5.00	5.17	5.11
	2379	42713609	46134033	46100037	227.07	0.14	5.00	8.01	7.93
Average	2835.00	44161388.0	47758468.0	47681312.0	202.87	0.13	5.00	7.74	7.60
70000	7062	n/m	75626571	75552711	–	0.19	5.00	–	–
	6214	n/m	74843077	74762329	–	0.19	5.00	–	–
	3822	n/m	65062124	64988624	–	0.17	5.00	–	–
	6491	n/m	73271387	73220405	–	0.19	5.00	–	–
Average	5897.25	–	72200792.0	72131016.0	–	0.19	5.00	–	–
90000	8618	n/m	97054560	96997926	–	0.25	5.00	–	–
	474	n/m	66596853	66589769	–	0.23	5.00	–	–
	9688	n/m	102440048	102380832	–	0.25	5.00	–	–
	1079	n/m	68548146	68540540	–	0.23	5.00	–	–
Average	4964.75	–	83659904.0	83627264.0	–	0.24	5.00	–	–

Table 3.IV – Comparison of Exact, Greedy and RVNS methods on large size instances with m=50 parking lots, dummy parking and different number of vehicles n; 'n/m' - no memory.

our min-sum-sum parking allocation model is *"integer friendly"* and therefore not NP-hard. However, for very large and more realistic sizes (e.g., for $n \geq 30000$), reaching the optimal solution is not decisive, either because of the time to reach it is unpredictable and too long, or due to memory overflow. Our basic model is static, but it can cover the dynamic nature of the problem by repeating its execution very often, every five seconds, for example. Therefore, it is more important to get an approximate solution fast within a fixed time limit, rather than to get an exact one in unpredictable time. To guarantee that a good quality solution is obtained in each time step, we developed a VNS-based

heuristic. Computational results on randomly generated test instances demonstrate that the exact solution approach is better on smaller instances, but for larger ones, the heuristic approach is more reliable because its stopping condition is the maximum execution time for the search.

This static model will represent the tool for assigning parking lots to vehicles within a dynamic configuration. In the next chapter, the static PAP will be solved over time and will guarantee the optimality for that time moment. However, as will be presented in the following chapter, this does not guarantee the optimality of the overall (dynamic) problem. Therefore, a mechanism which deploys the static PAPs and evaluates it is required. In Chapter 4 we present a framework capable of tackling such changes in near real-time.

CHAPTER 4

DYNAMIC PARKING ALLOCATION

In the previous chapter we showed that the static PAP can be modeled with a simple Boolean LP and quickly solved. However, its results don't have practical usefulness, due to the ever-changing input. More precisely, if parking lots are assigned to vehicles at some point of time, in the very near future there can be significant modifications in traffic: the number of vehicles, the number of cancellations, traffic congestion, etc. These changes in traffic require adapting, thus a framework capable of adjusting to them is necessary. This chapter addresses the dynamic aspect of the PAP by introducing a framework capable of providing real-time parking allocations to a large set of vehicles. The framework is based on sequentially solving a static PAP over a given time horizon, i.e., an online algorithm. The transition from one decision moment to the other requires a mechanism which acts in-between them. Rather than devoting ourselves to include various stochastic variables that represent some uncertain changes of input, we make use of Property 3 of the static model proposed in Chapter 3, so that if any unexpected change appears it we will be able to reassign new parking lots to vehicles in near real-time. The proposed framework relies on sequentially solving the static PAP model introduced in Chapter 3. Therefore, since the solution for the static PAP is quickly produced, we focus on the dynamic properties that are specific to the PAP, while keeping the requirements realistic. This produced a simple framework that can be easily modified to deal with different demands of the users and decision makers. Moreover, as will be demonstrated in Chapter 5 it is scalable and responsive, as it can handle huge sets of vehicles over a given time horizon in short periods of time. All of these properties make it flexible to adapt to various particular cases of interest. Moreover, this framework is not limited to the parking problem, but can be applied to most online dynamic problems.

Chapter outline This chapter starts by introducing several definitions of dynamic formulations found in the literature in Section 4.1. Moreover it goes into more detail of how a dynamic online problem can be stated as well as the solution strategies. Section 4.2 introduces the framework we developed for the PAP, called the DPAP framework. Beside a dynamic mechanism that sequentially solves the static PAP at each cycle, the DPAP framework also includes layers that make it more flexible to be incorporated in practical terms. Namely, the real data and policy layers. The next section shows how the model presented in the Chapter 3 can be applied within the DPAP framework, while Section 4.4 presents our implementation of the proposed online algorithm. Section 4.6 offers some concluding remarks to this chapter.

4.1 Dynamic combinatorial optimization problems

The first endeavor of this chapter is to introduce a dynamic form of an optimization problem. As mentioned in Chapter 2, there is no clear way to formulate a dynamic problem. However, we observe that most authors opted for a sequential approach, by solving one static problem at a time. In this section, we first more precisely define a dynamic formulation, then adapt the model (3.2)-(3.6) from Chapter 3 to include the decision moment, i.e., to place each static PAP within a planning horizon.

Notation and terminology

In the context of optimization problems, a problem is considered to be dynamic when data are gradually received over a given planning horizon. Therefore we can say that a dynamic problem represents a sequence of input updates that can be solved over a given period. The points in time of the planning horizon for which the problem is solved are called the *decision moments* or *decision points*. At this fixed time t, a static problem is solved with the currently available input denoted by W_t. The state of the dynamic problem at time t is denoted by S_t, and is called *state variables*. It is assumed that the state S_t contains all the needed information to model our system at any future time point $l > t$ of the planning horizon. This includes the case where some parameters are probabilistic. However, a dynamic problem is not just sequence of static problems. It encompasses also the transitions between these decision moments.

Let us denote by $\mathscr{I}$ a dynamic problem and by $H = \{t_1, t_2, \ldots, t_h\}$ a discretized planning horizon, where each element $t_k, k = 1 \ldots, h$ is called a *time step*. The set of decision moments denoted by $\Delta = \{\delta_1, \ldots, \delta_N\}$ is a subset of H. Then the dynamic problem $\mathscr{I}$ can be viewed as a progression of static problems denoted by $I_1, \ldots, I_N$ at N decision moments, where each problem $I_k, k = 2, \ldots N$, is defined by its previous state variables S_{k-1}, input W_k and solution χ_{k-1}, i.e., $I_k = I_k(S_{k-1}, W_k, \chi_{k-1})$. Once the decision χ_k is made and new information W_{k+1} revealed, a new state S_{k+1} is obtained by a *transition function* ϕ, i.e.

$$S_{k+1} = \phi(S_k, \chi_k, W_{k+1}). \tag{4.1}$$

Therefore the goal is to solve $\mathscr{I}$ for a specified exogenous information process, consisting of the sequence

$$(S_0, W_1, \ldots, W_N). \tag{4.2}$$

Note that the verb "solve" is not clear-cut as in static deterministic problems. Determining the best decision, also called *decision rule* or *policy* given the currently accumulated information is relatively straightforward. However, if the aim is to take into consideration the upcoming input $t+1$ at the decision moment t then such policies are referred to as *lookahead policies*. That is why in [87] Powell proposes a unified framework for stochastic optimization (the author implicitly considers a stochastic problem to be dynamic) and underlines two types of challenges: modeling a sequential decision processes (state variables, input and decisions) and designing policies. Designing policies in our context can be viewed as the objective (cost) function of the overall dynamic process, which corresponds to (Q2) from Chapter 2. In the following subsections we first take a look to what a dynamic problem is optimizing in context of stochastic optimization. Deterministic optimization problems can always be considered as a special type of stochastic ones, thus we shall not consider them apart.

Solution strategies

At each decision moment $k \in \Delta$, the current decisions are evaluated by a *performance metric* or in classical terms an objective function denoted by $F_t = F_k(S_{k-1}, \chi_{k-1}, W_k)$. In the context of stochastic optimization the goal is find a decision rule π to maximize (or minimize) the following formula

$$\max_{\pi} \mathbb{E}\left\{ \sum_{t=0}^{N} F_t(S_t, \chi_t^{\pi}, W_{t+1}) | S_0 \right\}, \tag{4.3}$$

where $\mathbb{E}$ is the expectation function for the random variable W of the sequence (4.2). Equation (4.3) is called the *cumulative reward*, because it sums all the benefits (or costs in the case of minimizing) of decisions χ_t, $t = 1, \ldots, N$. In Powell [87], the authors argues that the (stochastic) optimization problem (4.1)-(4.3) can represent any expectation-based (stochastic) optimization problem and proposes a unified framework for stochas-

tic optimization problems. While cumulative reward takes into consideration the overall decisions taken oven N decisions, the *terminal* or *final reward* is referred to only the solution at the decision moment N, i.e., at the end of the time horizon. The terminal reward in Powell's unified framework is formulated as follows:

$$\max_{\pi} \mathbb{E}\left\{C_N(x_N^{\pi}, \widehat{W})|S_0\right\}, \tag{4.4}$$

where x_N^{π} is a final decision that we achieved following the policy $X_t^{\pi}(S_t)$ observing the input or training observation $W_1, ..., W_T$ and then evaluate over the random variable $\widehat{W}$.

In this manuscript we do not search to minimize the expectation function of a sequence of random variables (W), nor to explicitly find the best policy by solving (4.3). We will define policies differently, and empirically test them over a sequence of PAPs. Moreover, we focus on the cumulative reward and solve the dynamic problem as an online problem. The terminal reward cannot directly be compared to the cumulative one, but can indicate on the quality of its solution.

The main reason we opted for such a setup for the PAP is because our main goal is to assign parking lots to vehicles as quickly as possible. If it can be solved as new data is revealed (data-driven), then it can compensate for the shortage of future data. Another way of looking at it can be that, if the static problems are solved quickly, this can alleviate the dynamic problem of most of its probabilistic input. In other words, we can keep the number of stochastic variables low. Furthermore, we are more interested in the operational decision making (short-term), and searching for the best policy function, and minimizing the expectation would imply a more strategic approach.

Online dynamic problems When solving a dynamic problem $\mathscr{I}$ as an online problem, i.e., data-driven, then it has its corresponding static problem I. The static problem I is in fact the equation (4.3) where W are known and for a fixed policy function π. Denote by F an objective function of $\mathscr{I}$ and by f an objective function for I, i.e., a function that would have been used had all the data been available. The aim is to solve $\mathscr{I}$ so that at the end of the planning period the solution achieved is as close as possible to an optimal solution x of I with respect to f. The problems $\mathscr{I}$ and I cannot be directly compared,

but f can still provide an estimate of a good solution of the dynamic counterpart F. Over a planning horizon H with N decision moments, there would be N static subproblems $I_1,\ldots,I_N$ of the problem $\mathscr{I}$, where I would correspond to I_N. Solving each subproblem optimally, does not necessarily yield the optimal solution of $\mathscr{I}$. The optimal solution $\dot{x}$ of a subproblem $I_k, k = 1,\ldots,N$, of is called a *tentative solution*, while the solution that is implemented during that time interval is called *permanent solution*.

In this manuscript, every allocation made to a vehicle is a tentative solution, until it is close enough to its designated parking lot. Each subproblem I_k is solved optimally (and approximately, for testing purposes) until the last decision moment. The choice of the decision moments, and the way we organized the dynamic allocation are detailed in the following sections.

This type of online mechanism can be applied to a variety of dynamic problems. More precisely, any problem which is based on a optimization problem with continuously arriving input. Moreover, the dynamic PAP framework, that will be presented in the following section, can also be generalized to other types of problems, and is not limited to parking assignment.

4.2 Dynamic PAP framework

In order to implement an online algorithm for the PAP we first need to provide a framework that can handle the changes of state (S_t) over time t. We call this framework the *DPAP framework* and this section will lay out it effectiveness: scalable, responsive and flexible.

The DPAP framework is structured in four layers: (i) the policies which determine the dynamic setup, (ii) the dynamic PAP, (iii) the MIP model for solving the PAP, and (iv) the collected data (Figure 4.1). The layers (i) and (iv) answer the question (Q1) from Chapter 2.2 and represent the infrastructural requirement of the classification proposed in Section 2.5 of Chapter 2. The layer (ii) answers the question (Q3) and corresponds to the dynamic aspect of the classification, while the layer (iii) answers (Q2) and corresponds to the static aspect of the PAP. Note that, the policies of our framework are not functions as in stochastic optimization problems, but rather a set of strategies that would restrict drivers from some undesirable parking lots. These policies are evaluated empirically,

rather than analytically as in (4.3).

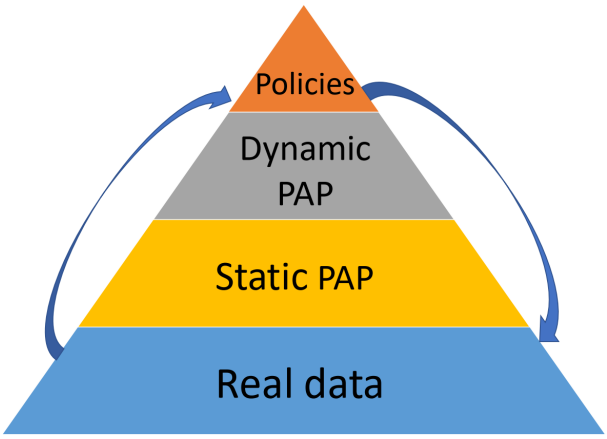

Figure 4.1 – DPAP framework

Layer (i): policies In the context of the DPAP framework, the policies determine the decision moments and the subset of parking lots to which the dynamic PAP (DPAP) will be applied. More precisely, in a dynamic setup the choice of the set of decision moments represents the backbone of the dynamic process. In other words, the choice of a subset of all the time steps to represent the decision moments. A special case of the choice of decision moments would be if the only decision moment is at the end of the planning horizon. This would then correspond to the case of the static problem I where all the data are known. On the other hand, every time step of the planning horizon can be considered as a decision moment. This can only be possible if we can guarantee that an assignment can be made to all the vehicles before the next decision moment. This would coincide with the continuous updates of vehicles requests for parking. Therefore it can be said that the most challenging case of the PAP is where the decisions are made as soon as possible. The second role of the policies is to take into account that there could be additional rules or regulations that would make certain assignments unfavorable or even infeasible at a given moment.

The policy layer is separate from the other layers of our framework, because it is independent from them, i.e., the policies do not change the complexity of the other layers. It sets rules that have minimal computational effort that provide guidelines for

subsequent layers. The policies that were used in our experimentation are detailed in Chapter 5.

Layer (ii): dynamic PAP Once the guidelines are defined in the form of policies, we can proceed with a mechanism that can ensure an effective dynamic allocations sequence. As we shown in Chapter 3, the static PAP model (3.2)-(3.6) can be solved optimally in a short period of time. This fact leads us to believe that an online algorithm could prove to yield good results. This online algorithm is the basis of the DPAP layer. It refers to the mechanism that handles the dynamic nature of the problem, i.e., the continuous updating of vehicles and their large number in real time. One of the main functionalities of the online algorithm is to follow the state variables. Aside from the state variables, the mechanism is also defined by several parameters that ensure a smooth input inflow. Section 4.3 illustrates in detail the full implementation of this mechanism.

Layer (iii): static PAP The static PAP is the tool that allocates parking lots to vehicles. It is defined by means of a 0-1 programming model and can be solved both exactly and approximately. By design, the model is made to be totally unimodular, making it easy to solve with exact methods. Real data are used to determine some parameters of the previous layers, thus ensuring that our framework is fed with accurate historical data.

Note that the static PAP can be any combinatorial optimization model and not necessarily one that is TU. For example, it can be a variation of one of the models presented in Chapter 3, or a VRPTW ([91]), or a TVTSP ([110]). However, as mentioned in [100] we search for a balance between the simplicity of the 0-1 model and the wealth of the dynamic elements and opt for the (3.2)-(3.6) as the basis of our static PAP. It obviously requires some minor changes, but it still keeps its properties.

Layer (iv): real data Layer (i) and (ii) depend on the traffic information of the city. More precisely, to formulate a more rigorous and realistic problem we make use of publicly available parking data. Most cities dispose of real-time parking availability and some publish them over the internet. We use these data to estimate the number of vehicles and the availability over time.

As the policy layer, this layer is independent from the others and does not influence

the overall complexity or stability of the framework. However, it will be used to determine some of the parameters that will be used in other layers, such as the estimation of the state variables.

4.3 Online DPAP

In this thesis we propose an online algorithm to tackle the PAP. In other words, to sequentially solve the static PAP, as new data are revealed. In Chapter 3 we introduced the static PAP model, in this section we modify it to adhere to the online algorithm. Moreover, we introduce the notation that will be used and provide the specificities of the dynamic variant (see [73, 76, 77]).

Assumptions & notation

Let $H = \{t_1, \ldots, t_h\}$ denote the discretized planning time horizon $[0, T]$, where each element $t_k, k = 1, \ldots, h$ is called a *time step*. Further, let $V^k = V(t_k)$ and $P^k = P(t_k)$ be the set of vehicles and the set of parking lots at time step t_k, respectively ($k = 1, \ldots, h$). Note that the set of parking lots will not change over time, i.e., $P^k = P$, and we denote its number with $m = |P|$. We assume that the current positions (origins) and the destinations of all the vehicles $i \in V^k$ are known. Each parking facility $j \in P$ has a capacity q_j. The total capacity q_j alone cannot guarantee an available slot at the arrival time of a vehicle. Hence, we include a fluctuating *residual capacity*, at each time step $t_k \in H$ for every parking j, denoted by c_{jt} ([75]). The time needed for the vehicle i to arrive at parking j is denoted by t'_{ij}. We also include the time that the driver i would spend to move from its designated parking j to its destination, denoted by t''_{ji} (see Figure 4.2).

Invariants We recognize that some values will remain unchanged over time, and will not be influenced by the framework. The most obvious invariant is the number of parking facilities $m = |P^k|, P^k = P, k \in \{1, \ldots, h\}$, and their total capacities q_j, which do not change over time. Although a parking lot may close during the day (as in our collected data), this situation is covered with the residual capacity parameter c_{jt}, which will take the value of 0, during that period. Since we consider that the vehicles will not change their destinations, we assume that the traveling times from their parking to the their final

destination (t''_{ji}) will also be constant over the planning horizon.

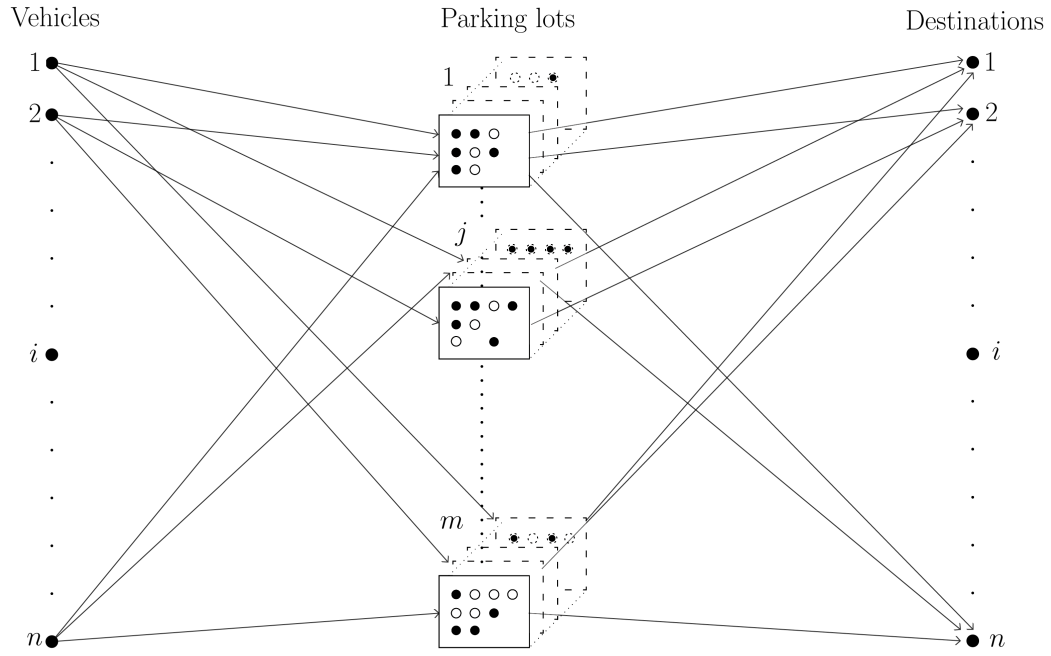

Figure 4.2 – Representation of the DPAP for one decision moment.

The left-hand side circles of Figure 4.2 represent the current locations of the vehicles. The rectangles in the middle represent the parking lots, at different time periods with their residual capacity c_{jt} depicted with white and black circles. The right-hand side circles correspond to the vehicles destinations. Note that the value of t'_{ij} is the time the vehicle needs to arrive at a parking $j \in P$, but its arrival clock time will be at the time step $t_k + t'_{ij}$, in the overall time planning horizon T.

To tackle the dynamic aspect of the problem, we introduce the set of vehicles that have arrived at their assigned parking denoted by V_a. The number of time steps a vehicle i has spent in our DPAP framework before reaching its lot is denoted by τ_i. Finally, f^{DPAP} returns the value of DPAP by calculating the cumulative time all the vehicles have spent in the system.

The policy Π will determine the subset of potential parking facilities, per vehicle i, denoted by $P_i = P_i(\Pi)$. This prevents an unfavorable allocation for each vehicle i. The policies also define the decision moments $\delta_k \subset H$ at which the DPAP will be deployed.

	Notation	Size	Time-dependent	Definition
Sets	H	$\|H\| = h$	✗	Planning horizon
	V^k	depends on k	✓	Set of vehicles at time step k
	P	$\|P\| = m$	✗	Set of parking lots
	$P_i(\Pi)$	depends on policy Π and vehicle i	✗	Subset of potential parking lot of vehicle i
	V_a	depends on k	✓	Set of vehicles arrived to their parking lot
	Δ	$\|\Delta| \leq h$	✗	Set of decision moments
Parameters	t_k	$t_k \in H$	✗	k^{th} element of the planning horizon, called k^{th} time step
	$C = (c_{jt})$	$m \times h$	✗	Residual parking capacity
	$T' = (t'_{ij})$	$\|V^k\| \times m$	✓	Traveling time of vehicle i to parking lot j
	$T'' = (t''_{ji})$	$m \times \|V^k\|$	✓	Walking time from parking j destination i
	τ_i	$\|V_a\|$	✓	Time vehicle i spends in the system

Table 4.I – Notation summary

To demonstrate the responsiveness and flexibility of our framework we set each time step to be a decision moment $\delta_k = t_k$. The notation is summarized in Table 4.I.

0-1 programming model PAP(k)

For a given step t_k of the planning horizon H, we denote the static model by PAP(k) = PAP(t_k). The 0-1 integer programming formulation of the PAP(k) uses binary variables x_{ij} equal to one if and only if parking j is assigned to vehicle i. The model can be stated as follows:

$$\text{minimize} \sum_{i \in V^k} \sum_{j \in P} \left[t'_{ij} + t''_{ji} \right] x_{ij} \tag{4.5}$$

$$\sum_{j \in P_i} x_{ij} = 1 \qquad i \in V^k \tag{4.6}$$

$$\sum_{i \in V^k} \alpha_{ij}^t x_{ij} \leq c_{jt} \qquad j \in P, t \in H \tag{4.7}$$

$$x_{ij} \in \{0,1\} \qquad i \in V^k, j \in P, \tag{4.8}$$

where

$$\alpha_{ij}^t = \begin{cases} 1 & \text{if } t = t'_{ij} + t_k, \\ 0 & \text{otherwise.} \end{cases}$$

The objective (4.5) is to minimize the total traveling time of all vehicles, from their current position to their parking (t'_{ij}), including the traveling time to reach their destination from their assigned parking lot (t''_{ji}). Constraints (4.6) ensure that a parking will be assigned to each vehicle $i \in V^k$ within the set of potential parking facilities P_i determined by policy Π. Constraints (4.7) mean that the number of allocated vehicles to a parking j will not exceed the current capacity c_{jt} at the arrival time $t'_{ij} + t_k$. Note that in the dynamic case we should not impose the total capacity constraint, as in (3.4). This is because this constraint would limit the number of vehicles that can be parked during the period H by $\sum_{j=1}^{m} q_j$, i.e., $\sum_{k=1}^{h} |V_k| < \sum_{j=1}^{m} q_j$. This is obviously not the case, because during the horizon H vehicles arrive and leave, thus liberating some of the previously occupied slots. However, since we do not assume control over the entire set of vehicles in traffic, we cannot impose flow constraints. In other words, we do not impose that vehicles that enter a parking lot must exit it. This information is implicitly found in the residual capacity parameter $C = (c_{ij})$, which is retrieved from the real data layer.

Even though we include the time steps in this formulation, the decision variable x does not have to be represented with three indices, as in multi-dimensional APs (see Section 2.1 of Chapter 2). This is avoided by introducing the parameter α_{ij}^t which ensures that the total number of vehicles that will arrive at the same time ($t = t'_{ij} + t_k$) will not exceed the capacity c_{jt} for some parking $j \in P$. Further, since we do not coordinate

all the vehicles in traffic, and we cannot impose flow constraints to the model. Similarly to (3.2)-(3.6) the model (4.5)-(4.8) possesses the integrality property. This implies that a linear programming solver can be used to solve it to optimality in near real time.

Dummy parking facility If the number of vehicles exceeds the capacity (Constraints (4.7)), there will be no feasible solution to the model (4.5)-(4.8). Therefore, we include a dummy parking facility $m+1$, $P = P \cup \{m+1\}$, with a large residual capacity $c_{m+1\,t}, t = 1,\ldots,h$. In the static case of Chapter 3 the coordinates of the dummy facility could be arbitrary, as long as the traveling time was high enough to assign this lot to a vehicle only if no other slot is available. In the dynamic case we cannot do this. Therefore, to avoid setting the arrival time $t'_{i\,m+1}$ at an arbitrary high value, we set the dummy parking to be at the vehicles destination. More precisely, the time required to reach the dummy parking $t'_{i\,m+1}$ will be computed as if $m+1 = \text{destination}_i$, for all $i \in V^k$, i.e.,

$$t'_{i,m+1} = \text{time}(current\_position_i, destination_i), \quad i \in V_k.$$

In this way, if no other solution can be offered, the vehicle will be assigned to its destination, as it would have been if it had followed the GPS. However, the walking time to the dummy parking, $t''_{m+1,i}$, is set to a high value. This penalization will then allocate a vehicle to the dummy parking if and only if no other parking place is available. Note that for all vehicles i, the dummy lot is included in P_i, i.e., $(m+1) \in P_i$, for any policy Π.

Performance metric The objective function (4.5) is used only at the decision moment k, i.e., it is only valid for PAP(k). However, it cannot be used in the following decision moment $k+1$, because the arrival times $t'_{ij}, i \in V^{k+1}, j \in P$ and the number of vehicles will be updated. Thus, the objective function (4.5) cannot be used to attribute a value to the DPAP. Hence, to quantify the value of the DPAP, the performance metric is calculated as follows:

$$f^{\text{DPAP}} = \sum_{i \in V_a} \left[\tau_i + t''_{j(i),i}\right], \tag{4.9}$$

where $j(i)$ is the parking assigned to vehicle i, $j(i) \neq m+1$, and V_a is the set of vehicles that have arrived at their assigned parking. As previously mentioned, τ_i denotes the time that vehicle i has spent in the system at the end of the planning horizon. More precisely, if the vehicle i sent its first request at time t_1 and reached its parking lot at time t_2, then $\tau_i = t_2 - t_1$. If $j(i)$ of the equation (4.9) is dummy parking $m+1$, then it is not taken into account. This is why when comparing different policies, we also take into consideration the number of unparked vehicles at the end of the planning horizon as a auxiliary performance metric.

Feasible solutions If at least one vehicle is allocated to a dummy parking lot, we will consider that solution to be infeasible. Such solutions must be evaluated differently than feasible ones (4.9), and exert different properties. Moreover, they represent the case of a saturated parking offer/demand scenario which is of particular interest in practical terms.

4.4 Solving the dynamic parking allocation problem

Our solution for the DPAP is based on the following approach: (i) solve the static PAP at h decision moments over a planning interval T (e.g., 24 hours) discretized into h time steps; (ii) avoid conflicting solutions obtained by PAP(r) and PAP(s) ($r < s$), i.e., the case where two or more vehicles are allocated to the same parking slot at the same moment. To this end, all vehicles not yet parked are considered active in all subsequent runs, until reaching their parking. Hence, this allows changes of allocations in future runs, and does not limit a vehicle to reserve only one spot determined after the first request. This section illustrates our implementation of an online algorithm to solve the DPAP. Note that the DPAP in this case corresponds to the layer (ii) of the overall DPAP framework.

Conflicting solutions

Solving the dynamic PAP as a sequence of static problems may produce conflicts between two solutions obtained at different decision moments t_r and t_s ($r < s$). If we consider every set of vehicles V^k ($k \in H$) independently, we cannot guarantee that a future vehicle will be allocated the same slot at the same arrival time. To avoid this

conflict, at each decision moment, we take into consideration every vehicle that has not yet reached its parking. Therefore, the set of vehicles V^k to which we allocate parking lots will be

$$V^k = (V^{k-1} \setminus V_a^k) \cup V_{\text{new}}^k, \tag{4.10}$$

where V_a^k and V_{new}^k represent the vehicles that have arrived at their parking at time k, as well as the new vehicles appearing at time k, respectively. By keeping all the vehicles in the model, this allows us (i) to wait and allocate them a lot in a future time step if no such allocation can be made at time t_k; (ii) to avoid conflicts, i.e., situations in which two or more vehicles are allocated to the same slot; (iii) to change the allocated parking depending on the circumstances.

Dynamic PAP algorithm

The previous section introduced the Boolean MP model which will be used in the dynamic setup. In this subsection we present the concrete implementation of our online algorithm. The steps of our DPAP(h) are presented in Algorithm 5:

Algorithm 5 Dynamic parking allocation routine

Function DPAP (P, C, h)
Initialization
$V_a = \emptyset$; ▷ The list of arrived vehicles is initialized
$V = \{V_1, \ldots, V_n\}$; ▷ The set of first n vehicles
Compute $T'(V), T''(V)$; ▷ Matrices with road time estimations
$x \leftarrow \text{PAP}(n, T', T'', C)$; ▷ Get the initial solution x
Simulation loop
for $k \leftarrow 2$ *to* h **do** ▷ For all decision moments k
    Update(V, x) ▷ Updating vehicles positions toward their assigned parking facility
    $V_a \leftarrow V_a \cup V_a^k$; ▷ Based on the update, add vehicles arrived at time k
    $V \leftarrow V_{new} \cup (V \setminus V_a)$; ▷ New active vehicles are added and parked ones removed
    Compute $T'(V)$ and $T''(V)$
    $x \leftarrow \text{PAP}(n, T', T'', C)$; ▷ Solve the PAP with updated input
end for
return $f^{\text{DPAP}}(V_a)$ ▷ Compute the DPAP value

Initialization The DPAP Algorithm 5 requires three input parameters: the set of parking lots P, where $|P| = m$, the residual capacity matrix $C = (c_{jt})$ and the number of decision moments h. The first two are fetched from the real-data layer of the framework, while the number of decision moments is determined in the policy layer.

We receive the initial set of requests, denoted by V, which contains vehicle positions and destinations. According to their coordinates we compute the time needed to reach each parking $\left(T' = \{t'_{ij} : i \in V, j \in P_i\}\right)$, and from each parking their destinations $\left(T'' = \{t''_{ji} : i \in V, j \in P_i\}\right)$. We then solve the PAP(1) (4.5)–(4.8) for the given input.

Simulation loop Once the solution of the static PAP(1) has been computed (line 5), we update the coordinates of the vehicles. Vehicles that have arrived at their parking lots are stored in the list of parked vehicles V_a (line 8). At each decision moment k of the planning horizon H we receive new requests, which are added to the list of active vehicles V (line 9). The travel time matrices, T' and T'', are then computed for the vehicles remaining in the active list (line 11). Note that in this particular case we are solving the static model as presented in (3.2)–(3.6), but the same can be applied for any model that possesses the integrality property, such as the extensions presented in Section 3.2. Parking lots are allocated to the list of active vehicles by solving the model (4.5)–(4.8) (line 11). This procedure is repeated $h-1$ times, for each time step t_k. The resulting value of the DPAP f^{DPAP} is calculated as presented in (4.5), for which $S = V_a$ (line 13).

Figure 4.3 depicts the flowchart of the DPAP procedure (Algorithm 5). The blue box represents the real data which provide the parking information, the residual capacities and the planning interval. The grey boxes represent the phases of the DPAP layer, while the yellow boxes represent the static PAP.

Data structure The execution time of PAP(k) is crucial to our approach. It is strongly influenced by the data structure and the updating which we use in our implementation. The class `Vehicle` is defined by 10 attributes:

1. vehicle ID,
2. initial coordinates,
3. current coordinates,

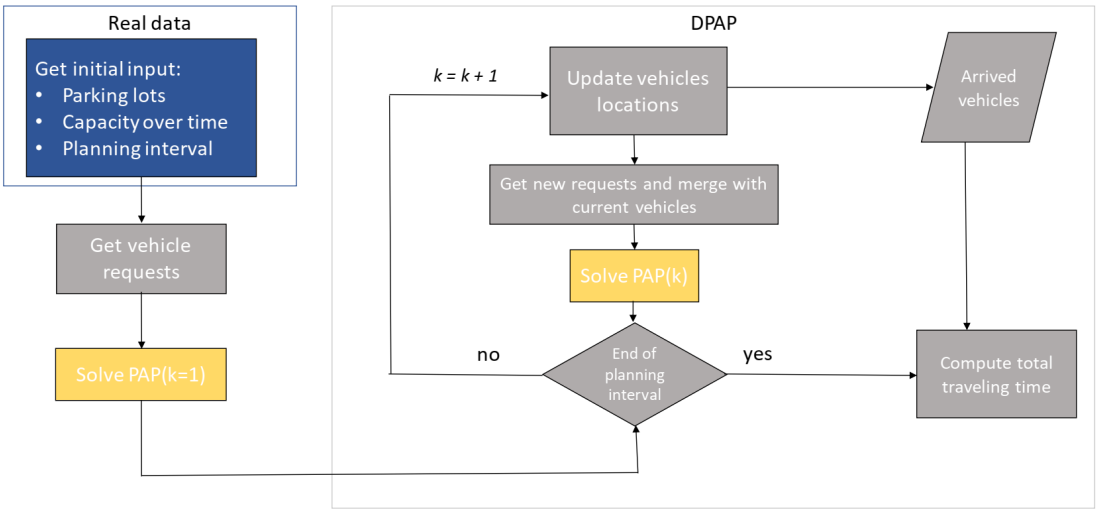

Figure 4.3 – DPAP flowchart

4. coordinates of the destination,
5. currently allocated parking,
6. previously allocated parking(s),
7. the decision moment when the vehicle first sent its request,
8. the decision moment when the vehicle arrived to its parking,
9. number of changed assignments,
10. number of times that time was lost, per change.

The vehicle requests are stored into a list, `active_vehicles`, until they have reached their parking. Once they arrive, they are copied into the list of allocated vehicles, `arrived_vehicles`. The traveling time matrices T', T'' are calculated based on the current coordinates and the currently allocated parking attributes. We suppose that the drivers are driving at a constant speed of 30 km/h, and have a constant walking speed of six km/h. The speed can be an additional (variable) attribute, which would then influence the arrival times t'_{total}. Adding new attributes does not change the complexity of the DPAP.

Time discretization The overall planning horizon H must be discretized in order to solve the MP model. This is because each assignment is made for a specific arrival time for which the current residual capacity must be verified. However, the discretization time step may vary and can be set as a parameter. In this manuscript, the time step is fixed to one minute, since some instances need around one minute to be solved, but it can be further reduced (or increased), depending on the requirements (see Chapter 5).

Number of allocations of the same vehicle Our framework allows the change of solutions over time, although the solution at this decision moment will be optimal, some vehicles may lose time with this new allocation. However, our results show that these changes do not deteriorate the travelling time of individual vehicles.

4.5 DPAP mechanism

In the previous section we presented our online algorithm for solving the sequence of static PAPs that we identified as one of the layers of our framework. In this section we briefly formally state the DPAP mechanism. Namely, every dynamic problem is defined by its state variables, and their respective transitions between phases. We show that the DPAP can be described by their vehicle set, which in term depend on all the previously made decision and the newly revealed data.

Transition phases

This section presents how the DPAP framework handles the transitions between the decision moments. In other words, the state variables and their transition function ϕ. Firstly, as in the [89] taxonomy, we identify the vehicle requests to be the dynamic component of the DPAP and thus set them to define the state variables, i.e., $S_k = V^k$. The *vehicle request set* V^k is defined at each decision moment $\delta_k, k = 1\ldots,h$, by its cardinality, i.e., simply the number of requests that require a parking allocation at a given decision moment.

To avoid conflicting solutions, we keep the vehicles that have not yet arrived to their parking lot active in the vehicle request set as illustrated in (4.10). This then enables us

to state the simple transition function

$$\phi(V_{k-1}, \chi_{k-1}, W_k) = |V_{k-1}| + \left|V^k_{\text{new}}(W_k)\right| - \left|V^k_a(\chi_{k-1})\right|. \tag{4.11}$$

Note that the transition function depends on the previous solutions from which we obtain the set V_a^{k-1} of vehicles that arrived since the previous decision moment and on the newly revealed number of request at time k, i.e., $V^k_{\text{new}}(W_k)$, where W_k denotes the input at the decision moment k. In Chapter 5 we will test the optimal and approximate decisions χ_k. For the input W_k, we will make use of the real data that is available on the web sites of three cities in Europe.

Effects and parameters

As demonstrated so far, the complexity of the DPAP framework is based on the static model, i.e., layer (iii). The mechanism we propose has not influence on the overall performance on the framework. This remains the case even if we introduce additional parameters. In this thesis, apart the performance metric (4.9), we also count the number of changes of allocations and the number of unparked vehicles. This can help us to understand the behaviour of the dynamic setup, as will be demonstrated in Chapter 5.

Another parameter that can be taken into consideration is the time at which we notify the drivers of their assigned parking lot. More precisely, we assign a parking lot to all the vehicles at each decision moment, but we are not obliged to inform all the drivers. We can keep the option of waiting to inform the drivers at a convenient time of their parking lot.

In this manuscript we keep track of a vehicle until it reaches its parking or destination. However, we could keep all the vehicles active until the end of the planning horizon. By doing so, we would not raise the complexity of the DPAP, but would potentially have a memory overflow problem, as seen in Chapter 3, when the number of vehicles reaches critical scale.

4.6 Conclusion

Most dynamic transportation problems are treated as online problems. This mainly means that a sequence of static problems are solved over the planning horizon as new data are revealed. When considering the PAP, new parking requests arrive continuously, and solving it in an online fashion represents a logical choice. However, the drawbacks of online solutions is that they can produce myopic traps, i.e., thus provide a poor long-term solution, or yet could be time-consuming to solve quickly. We avoid the first issue by assigning vehicles to parking lots keeping in mind the parking lot capacity at their arrival times. Moreover, we keep the option of notifying the drivers when a good opportunity arises. The second drawback is solved by inteligently using the available data, more precisely the real-time parking availability, i.e., the residual capacity matrix C. This then allows us to make use of the model (3.2)-(3.6) from Chapter 3 which possesses the integrality property, and can be quickly solved to optimality.

As discussed in Chapter 2, the PAP should have a decision tool for assigning parking to vehicles, a dynamic setup that can enable the sequential deployment of the decision tool and an overall performance metric (see Section 2.3 of Chapter 2). In this chapter we answered all these points by introducing the DPAP framework. It is divided into four layers, which correspond to the policies, dynamic mechanism, static model, and real data. The policies are not functions as in SOP, but an umbrella of guidelines that can correspond to municipal regulations or user preferences. These policies provide more features to our framework without increasing the complexity. Moreover, they are empirically determined and not analytically. For the dynamic setup, we proposed a separate layer called the dynamic mechanism or DPAP. It ensures fluid transitions between each decision moment by keeping all the active vehicles in the system until they reach their parking lot. Moreover, we formally define the transitions and measure the number of changes occurred during the planning horizon. The DPAP is an online algorithm which in term solves the (4.5)–(4.8) at each decision moment, i.e., each cycle of the DPAP sequence consists of solving a static PAP. This static PAP is the decision tool that determines the complexity of the framework. As previously mentioned the model we opted for a model that possesses the integrality property, and thus renders the complexity at the lowest level, i.e., it represents a class of P problems in combinatorial optimization terms.

All these layers are fed with real data collected from publicly available sources to further improve to quality of our approach. These data are used to determine the transitions and the vehicle request set which then serve to form the parameters of the DPAP mechanism.

The framework developed in this chapter can be extended to other dynamic problems that are solved in an online fashion and not only the PAP. Furthermore, we believe that the simplicity of our approach represents its greatest strength. In the next chapter we evaluate our framework based on data collected from Belgrade, Luxembourg and Lyon.

CHAPTER 5

SIMULATION ENVIRONMENT

This chapter's goal is to validate the DPAP framework proposed in Chapter 4. The framework needs to be (i) responsive: it must provide good solutions in near real-time; (ii) scalable: it must be capable of tackling large number of requests for any given policy; (iii) robust: what ever changes are made it should provide stable (consistent) results. In order to test the responsiveness, scalability and robustness we have developed a simulated environment based on real data from three European cities: Belgrade, Luxembourg, and Lyon. Through publicly available data, we were able to collect the coordinates of parking lots and their real-time capacities. Based on these data we were then able to define maps for each city and conduct experiments of various policies of the DPAP framework. We overpopulate the vehicle request set to replicate the case when there is not enough parking slots for all the vehicles. Our results indicate that the dynamic aspect of the problem becomes most clear under these conditions because the most allocation changes are recorded. Moreover, the DPAP provided near real time solutions for 200,000 vehicles over the period of one day. The deviation in the results was low further indicating the robustness of our approach.

Chapter outline This chapter is organized into two main sections: Section 5.1 and 5.2. This first presents the details about the configuration for the DPAP framework, while the second presents the computational results. Section 5.1 provides details about the collected data and how the maps and vehicles are generated upon which a simulated environment was built to test our framework. Section 5.2 introduces concrete policies of the DPAP framework and presents the results obtained when applying these policies on real data. The concluding remarks of Chapter 5 are provided in Section 5.3

5.1 Real data and policies

To evaluate the DPAP framework we have simulated a real environment. The following section describes how the environment was set up and defines the parameters that were used.

Collected data

We found three cities with more than 500,000 inhabitants and with accessible real-time parking availability information online: Belgrade (Serbia) [1], Luxembourg (Luxembourg) [2] and Lyon (France) [3].

Belgrade disposes of 24 publicly operated parking lots with capacity ranging from 53 to 1,542 slots and a total capacity of around 10,000 parking slots over an area of 140 km^2. The city of Lyon reported 93 parking facilities (managed by different agencies) with a total of around 43,000 slots, covering around 450 km^2. Luxembourg makes 25 of its parking lots available and has the smallest area with around 60 km^2. The capacity span lies between 162 and 2,442, with a total of 8,067 parking slots. Data from Belgrade were collected every two minutes, while the refresh rate for Luxembourg and Lyon was three and four minutes, respectively. Note that the data for some parking lots in Lyon and Luxembourg were not always available.

With regard to the central theme of this thesis, parking allocation in peak hours, we set apart the weekdays and the weekends. We note (see Figures 5.1 and 5.2) that a similar behavior can be observed during weekdays when most of the traffic congestion occurs, while the weekends show less vehicle activity (Figure 5.2). During the data collection, the weekends did not include any event that would have yielded a significant increase of vehicle activity (e.g., football matches, severe traffic accidents, public demonstrations, etc.). In Section 3.2 we suggested several potential objective functions that could be used for the PAP and would keep the integrality property. One of them was the one that would include a preference to parking pricing. While collecting real pricing data and performing initial tests, we have concluded that the differences in parking pricing are too low do influence the final allocation decision.

1. https://parking-servis.co.rs/lat/gde-mogu-da-parkiram/
2. https://www.vdl.lu/fr/se-deplacer/en-voiture/parkings-et-pr
3. https://data.grandlyon.com/equipements/parking-disponibilitfs-temps-rfel/#data

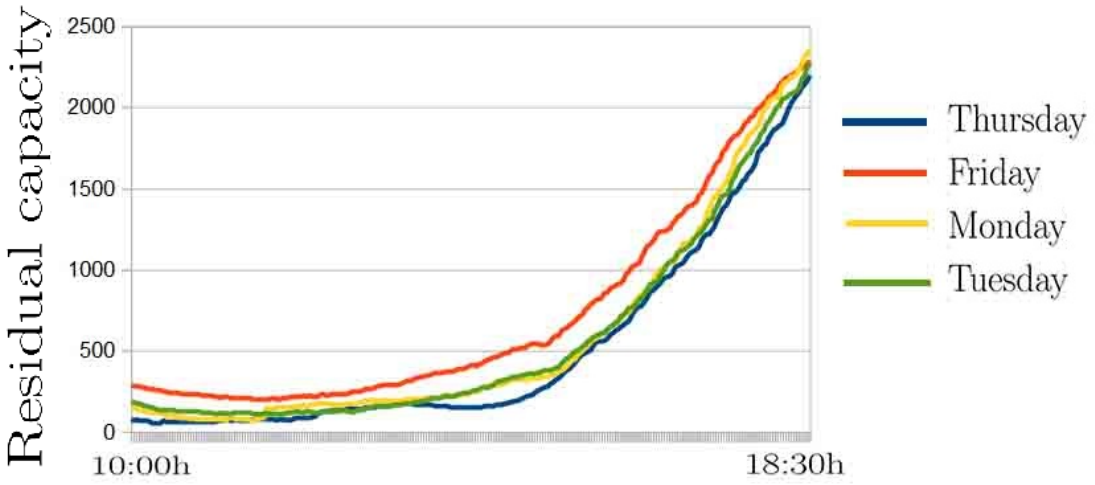

Figure 5.1 – Residual capacity change over time for March 2017 for the Bouillon parking in Luxembourg.

Filtered data Although the city authorities made the parking data publicly available, much of it was incomplete or contained errors. When data for some parking were corrupted or incomplete for a longer period of time, e.g., several hours, we excluded them from our tests. These filtered data were then injected to be the residual capacity parameter $C = (c_{jt})$, $j \in P, t \in H$. All the data, including parking availability, total capacity, geographical coordinates and detailed results are made available online: https://goo.gl/7JFhnt. This link also provides the unfiltered raw data that was collected.

Simulation parameters

We first introduce the random variables needed to simulate the dynamic parking process for the time interval T. The parameters that are discussed here, such as the number of vehicles appearing at each time step, their coordinates and destinations, are known in the real-world application. However, for the purpose of the simulation we need to introduce them as random variables. Each simulation starts at 00:00 and ends at 23:59 the same day, and is discretized into 1,440 one-minute time steps.

Time discretization and approximating the arrival times Most of the parking availability data found online is updated each minute. Moreover, updates for each parking

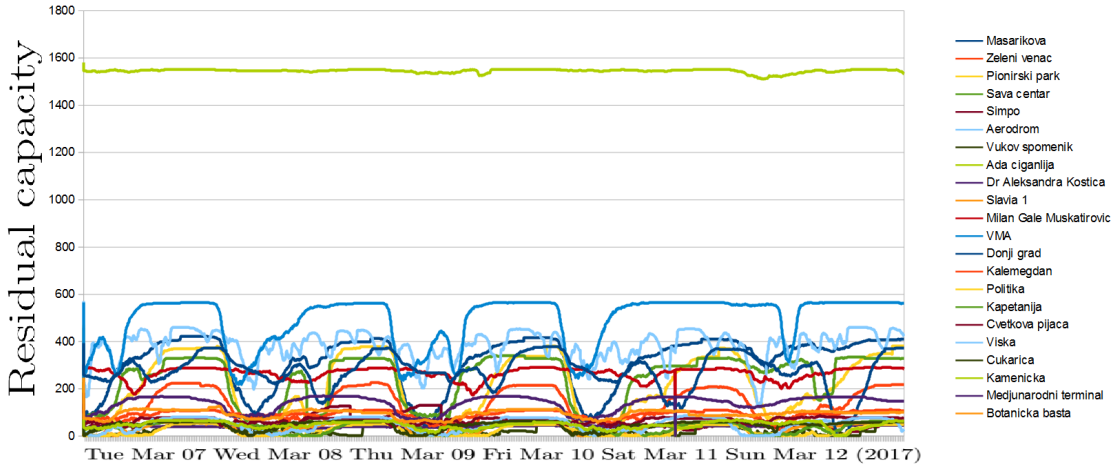

Figure 5.2 – Residual capacity change over time for all the parking lots in Belgrade, over several days in March 2017. The parking lot Ada ciganlija, represented in light green color, is mostly unused because it is the beach parking and the data was collected in the end of the winter.

lot are not synchronized, meaning that the availability of some parking lot is updated more frequently than others. That is why we collected these data each two, three and four minutes for the cities of Belgrade, Luxembourg and Lyon, respectively. In order to homogenize the discretization time step was set to one minute.

When computing the arrival times of vehicles (T'_k) at some decision moment k the values are real. This requires another step of rounding. However, we need to round these values to their ceiling, and not their closest integer because the vehicles will still be en route at that time step (see Figure 5.3).

In Figure 5.3 the first row represents the real travelling time of a vehicle. The second row represents the time steps of the planning horizon, while the third row represents the decision moments. The dotted red lines represent the time when the vehicle first sent its request, the time at which it arrived at its parking, and ultimately the time it at which exited our system, i.e., at the following decision moments. The time τ_i a vehicle i spends in our system if the distance between the entering and exiting decision moment. In the example presented in Figure 5.3 the vehicle i spent four minutes travelling, i.e., $\tau_i = 4$.

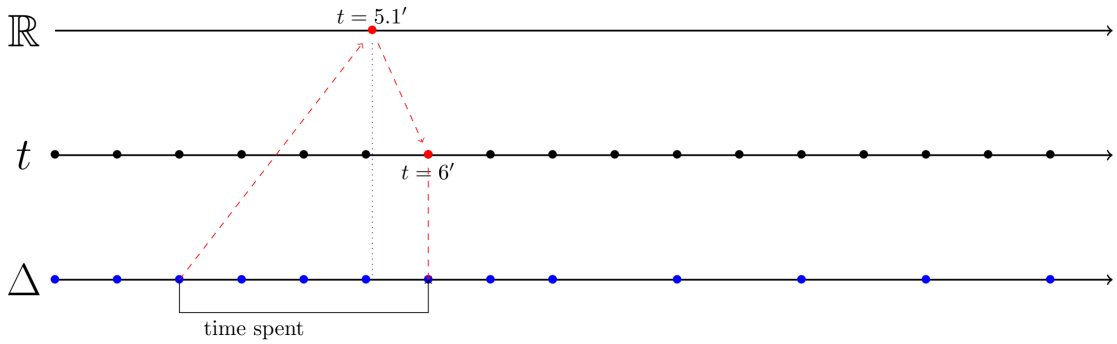

Figure 5.3 – Time discretization per vehicle

City map

We consider only the real geographical coordinates of both vehicles and parking lots. The coordinates are set in the two-dimensional spherical *latitude* (lat) and *longitude* (lon) coordinate system. For a given city, the set of parking lots is invariant and defines the boundaries of the area under study (see Figure 5.4 for the map of Lyon). The area is a rectangle R defined by two points, the most northeastern point NE, and most southwestern point SW as follows:

$$NE = (\max_{j \in P \setminus \{m+1\}} \{lat_j\}, \max_{j \in P \setminus \{m+1\}} \{lon_j\}),$$
$$SW = (\min_{j \in P \setminus \{m+1\}} \{lat_j\}, \min_{j \in P \setminus \{m+1\}} \{lon_j\}).$$

The center point C of this area, is computed as the arithmetical mean of all the parking areas, more precisely

$$C = (C\_lat, C\_lon) = \frac{1}{m} \sum_{j \in P \setminus \{m+1\}} \left(lat_j, lon_j\right).$$

The center C represents a good approximation of the real city center. This approximation was further confirmed to be accurate by examining the real city center (city by city) and comparing it with the values of C for each city.

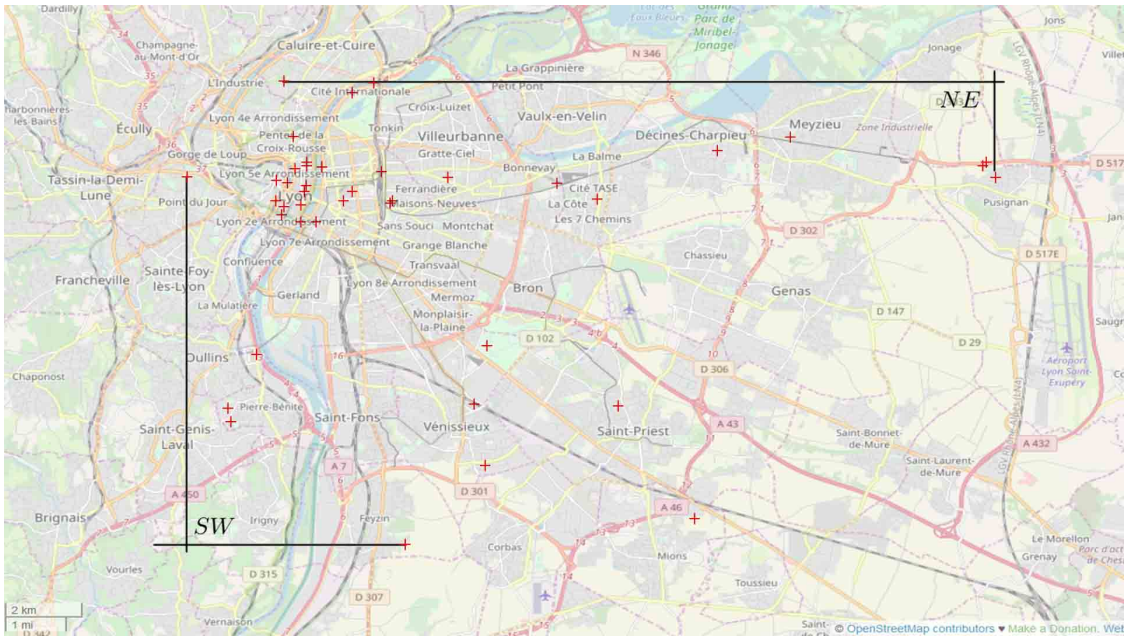

Figure 5.4 – Map of the city of Lyon. The red crosses represent parking lots.

Vehicle request set

As mentioned in Chapter 4 the state of our DPAP mechanism is defined by the vehicle request set, i.e., the set of requests received over a given time period. For each decision moment $\delta_k \in \Delta$, we have $n^k = n^{t_k}$ new vehicles appearing. However, all vehicles which have not yet reached their parking until time t_k still remain in our vehicle set V^k.

At each decision moment $t_k, k > 1$, the static PAP(k) will be fed with n_k vehicles, where

$$n_k = n_{k-1} + n_k^{(new)} - n_k^{(arrived)},$$

and where $n_k^{(arrived)}$ represents the number of vehicles which reached their parking at the decision moment t_k, obtained by PAP($k-1$).

Vehicle update At each step $k > 1$, before generating new vehicles and allocating them a parking lot, we verify which vehicles have arrived to their parking and we update their location. At each decision moment the vehicles coordinates are updated by a linear 2-dimensional motion toward their assigned parking lot.

Denote by $(lat_i, lon_i)^k$ the latitudinal and longitudinal coordinates of a vehicle i at time k and by s and d_i the speed and the distance between the vehicle's current coordinates and its parking lot, respectively. Let the coordinates of the parking lot j be denoted by $(lat_j, lon_j)^k$. Note that, the parking lot j can also be the dummy parking lot $j = m+1$, i.e., vehicle i destination. Let ∂ be the time elapsed since the last decision moment, i.e., $\partial = \delta_{k+1} - \delta_k$. Also let ML denote the rectilinear motion length, where $ML = \frac{\partial}{(s\,d_i)}$. The updated coordinates of the vehicle i at the decision moment $k+1$ will then be

$$(lat_i, lon_i)^{k+1} = ML(lat_j - lat_i, lon_j - lon_i)^k.$$

To verify if a vehicle has arrived to its parking lot is established by criteria: by the distance and by the time remaining to reach its parking. Namely, if the distance is less than 200 meters, or if its arrival time will be inferior to ∂, i.e., $t'_{ij} < \partial$, we consider the vehicle i has arrived at its parking lot j at the current decision moment k.

Vehicle generation To obtain a realistic approximation of the number of vehicles appearing during the planning horizon H, we rely on the real data layer of our framework. More precisely, we compute this value using the collected real data, i.e., the matrix $C = (c_{jt})$. We can assume that the number $n_k^{(new)}$ of vehicles appearing at some time $t_k \in H$ is proportional to the number of newly occupied slots of the recorded historical data, specifically,

$$n_k^{(new)} = \begin{cases} 0, & \text{if } \sum_{j \in P \setminus \{m+1\}} \left[c_{jt_k} - c_{jt_{k-1}} \right] \leq 0 \\ \left\lceil \gamma \sum_{j \in P \setminus \{m+1\}} \left[c_{jt_k} - c_{jt_{k-1}} \right] \right\rceil, & \text{else} \end{cases}$$

for some $\gamma \in \mathbb{R}^+$. Our assumptions are based on the historical data we accumulated over a period of several weeks in Belgrade, Luxembourg and Lyon (see Figures 5.1 and 5.2). When searching to reproduce these data with a discrete probability distribution function, the best fit was always a negative binomial distribution with two parameters. However, the obtained parameters did not allow to produce a good input.

We can set the parameter γ to a high value in order to observe how the DPAP will perform if the number of vehicles exceeds the number of available slots. The inflow of

requests, in the case of our data, is depicted in Figure 5.5. The vertical axis represents the number of requests, the horizontal axis represents the time, from 00:00 to 23:59, divided into 1,440 one-minute time steps.

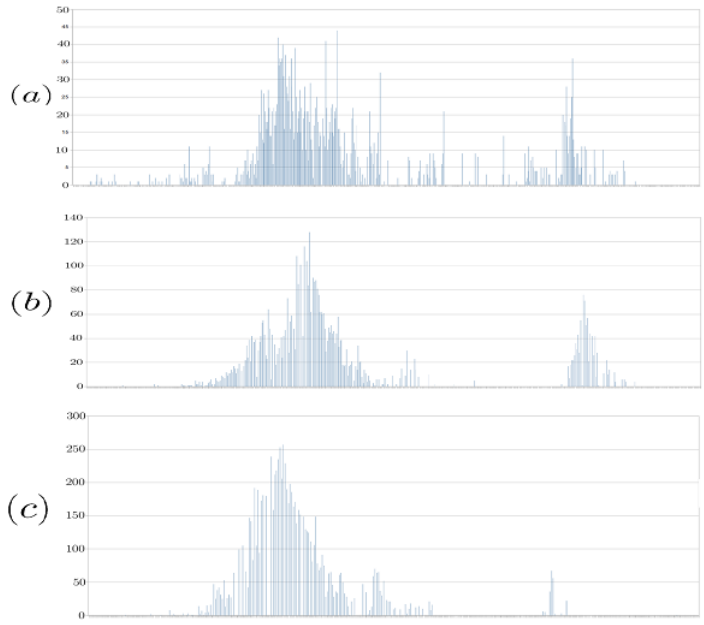

Figure 5.5 – Request inflow diagram for the cities of: (a) Belgrade, (b) Luxembourg and Lyon.

We set the vehicles speed to be constant at 30 km/h and the walking speed to be 6 km/h. In reality, both are time varying and endogenous, more specifically, depend on previous allocation. Extending the analysis to treat speeds that are time-varying and exogenous, and speeds that are endogenous would not influence the complexity of the DPAP framework. Therefore we just focus on constants values.

Vehicle location and destination

Once the vehicle number is set, we can attribute them a position and a destination. Let the set of positions be $\{(lat_i^-, lon_i^-)\}_{i \in V^k}$ and the set of destinations $\{(lat_i^+, lon_i^+)\}_{i \in V^k}$ at time k. We assume that vehicles can appear anywhere in the considered area R, with an equal probability, i.e.,

$$(lat_i^-, lon_i^-) \sim \mathscr{U}(R) \quad \forall i \in V^k \setminus V^{k-1},$$

where $\mathscr{U}(R)$ is a continuous uniform distribution on the area R.

Their destinations will be centered near the city center C. That is why we set the destination to be determined by a normal distribution with the mean being the center, and with a small standard deviation parameter, $\sigma = 0.15$, i.e.,

$$(lat_i^+, lon_i^+) \sim \mathscr{N}(C, \sigma^2) \quad \forall i \in V^k \setminus V^{k-1}.$$

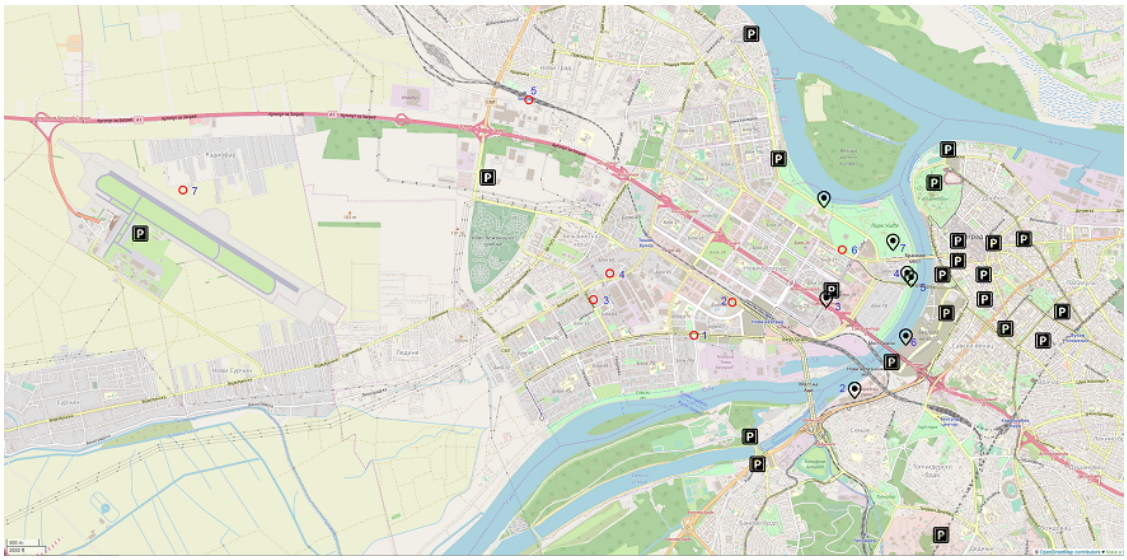

Figure 5.6 – An example of vehicle generation on the Belgrade map.

An example of generating seven vehicles on the map of the city of Belgrade is presented in Figure 5.6, where the red circles represent vehicles positions and the vehicles numbers, the pinpoints represent vehicle destinations. We can see that the center C is slightly west of the real city center. In each following time step $t_l, l > k$, the vehicles positions are updated in the direction of their assigned parking lot, until reaching it.

Policies

The policies are the strategic guidelines at the top layer of our framework introduced in Chapter 4. The main advantage and objective of these policies is to avoid assigning a distant parking to a vehicle just because no other solution can be found. This raises the

possibility of guiding a vehicle towards its destination (dummy facility), but keeps the model realistic. In this section, we consider a fixed decision time interval of one minute and three policies. The first policy depends on the vehicles' destinations, and the last two depend on the vehicles' current locations. In other words, since the destination will not change over time, the first policy is characterized by a fixed set of potential lots P_i for all vehicles i which will not change over time. The second and third policies depend on the vehicles' current location, which does change over the planning horizon H, and thus produces different sets P_i for each time step k.

Maximal walking time policy Π_1 We can impose a maximal distance the drivers are willing to walk from their allocated parking to their destination. This walking time can be introduced through a radius α_1 around their destination. For a given decision moment t_k this policy forms following subsets of P:

$$P_i(\Pi_1) = \{j \in P : t''_{ji} \leq \alpha_1\}, \quad i \in V^k.$$

Maximal traveling time policy Π_2 Similarly, we can set an upper bound α_2 on the total traveling time of a vehicle i. Note that these sets ($P_i(\Pi)$) depend on the decision moment k, i.e., $\Pi_2 = \Pi_2(k)$, and are defined as follows:

$$P_i(\Pi_2) = \{j \in P : t'_{ij} + t''_{ji} \leq \alpha_2\}, \quad i \in V^k.$$

Since vehicle i is guided towards a parking j, the traveling time t'_{ij}, for all $i \in V^k$, at the decision moment t_k, will be lower at each subsequent PAP$(l), l > k$. However, since a vehicle can be redirected, we can not guarantee that the set $P_i(\Pi_2)$ will enlarge during the travelling time of vehicle i.

Maximal deviation policy Π_3 A third option can be based on the individual perspective, i.e., the system takes into account that driver i does not want to deviate more than α_3 from its best parking. This forms the following subsets:

$$P_i(\Pi_3) = \{j \in P : t'_{ij} + t''_{ji} \leq \alpha_3 \min_l \{t'_{il} + t''_{li}\}\}, \quad i \in V^k.$$

In most cases the best parking will not change over time, but we cannot predict how $P_i(\Pi_3)$ will evolve over time.

5.2 Computational experiments

In this section we first compare the greedy heuristic with the exact algorithm. We then compare the policies introduced in Section 4. The computational experiments were coded in C++ Visual Studio 2012, executed on an Intel Core i7-4702MQ processor with 16GB RAM, running on a Windows 7 professional platform. CPLEX 12.6 was called via concert technology, coded in C++ on Visual Studio 2012 and ran in parallel on all cores, i.e., CPLEX default settings. For each setting of parameters, the simulation was run 10 times. We report the mean results in Tables 5.I-5.III. The execution time refers to time of the entire DPAP sequence, from PAP(1) to PAP(1440).

No-restriction policy

Even though the same set of new vehicles is added to the list of active vehicles at each time step, the CPLEX and greedy solutions can differ. As consequence, since we keep all the vehicles until they reach their parking, this will produce different input sets V^k for different time steps k. For example, at time step k a vehicle i can be allocated to parking j_1 by CPLEX, and to parking j_2 by the greedy algorithm. Both solutions produce the same total traveling time for vehicle i ($t'_{ij_1} + t''_{ij_1} = t'_{ij_2} + t''_{ij_2}$), but at time step $k+1$ the vehicle will not have the same coordinates for the CPLEX and greedy input values V^{k+1}. Over time these discrepancies can yield a significant gap. To mitigate this effect we split all the instances into two groups: those with standard capacities, and those with reduced capacities c_{jt}, and we remove the policy restrictions so that $P_i = P$ for any vehicle i. Note that in Table 5.I the objective f^{DPAP} is presented, while Table 5.II compares the number of unparked vehicles instead.

The greedy heuristic guarantees that no vehicle will remain without a parking at a given decision moment, if such an allocation is possible (Property 5), and it can deviate significantly from the optimal solution of a static instance (see Chapter 3. In this section we wish to examine how much deviation will the greedy heuristic error accumulate over the planning horizon. Its advantage remains the computing time (see Table 5.I), but as

will be show in Table 5.II its cumulative error leads to cases where not all vehicles find a spot, whereas the exact solution (for each decision moment) assigns to all the vehicles a parking slot.

			Execution time		**Number of changes**		$f^{\mathbf{DPAP}}$		
	m	$\lvert V_a \rvert$	CPLEX	Greedy	CPLEX	Greedy	CPLEX	Greedy	Difference
Begrade	23	3234	18.4	**0.1**	12.4	12.2	**68493.2**	68493.4	$< 0.1\%$
Luxembourg	18	6733	10.2	**0.05**	10.8	9.9	**88047.1**	88048.1	$< 0.1\%$
Lyon	47	10683	45.7	**0.8**	1233.5	1717.6	**279423.5**	280717.8	0.5%

Table 5.I – Instances with standard capacities where $P_i = P$ for all vehicles i.

From Table 5.I we observe that when there is sufficient parking capacity for all the vehicles, the greedy heuristic consitutes a better choice, because of its much smaller execution times. Nonetheless, the CPLEX execution times are at most 46 seconds, or 0.03 seconds per PAP. This confirms that this algorithm can be responsive in near real time. We also conclude that the number of changes is low, at most 0.16 per vehicle in the city of Lyon for the greedy heuristic.

			Execution time		**Number of changes**		**Unparked**		
	m	$\lvert V_a \rvert$	CPLEX	Greedy	CPLEX	Greedy	CPLEX	Greedy	Difference
Begrade	23	3234	20.0	0.1	879	1074	0	0	0%
Luxembourg	18	6733	15.7	0.07	5590.3	11467.6	0	81.7	∞
Lyon	47	10683	53.3	0.8	19918.2	31183.7	0	0	0%

Table 5.II – Instances with reduced capacities where $P_i = P$ for all vehicles i.

However, Table 5.II reveals that when the parking availability becomes limited, the exact algorithm presents clear advantages. More precisely, Property 5 (Chapter 3) guarantees that for the same input, the number of vehicles assigned to their destinations (the dummy parking) will be the same, but it does not guarantee the same solution. Therefore, over time the vehicle sets $V^k(greedy)$ and $V^k(exact)$ will diverge. This will lead to a different input for the PAP$(k+1)$. This then results in the greedy algorithm not being able to assign a parking to all vehicles. Such was the case for Luxembourg, where the average difference was 81.7. Furthermore, we encounter a significantly higher number of changes when comparing with the standard capacity. Namely, CPLEX recorded around 0.27 changes per vehicle for Belgrade, 0.83 in Luxembourg, and 1.87 for Lyon. This significant increase in the number of changes demonstrates how the framework adapts

in the event of low capacity. The execution times are slightly higher when capacities are reduced, but are still below one second per decision moment.

We have also created an example of traffic overload to assess the robustness and flexibility of our framework. On the larger area of Lyon and for reduced capacities, we set $\gamma = 20$, i.e., we multiplied the number of vehicles by 20 to produce a total of 213,660 vehicles. The average CPLEX execution time was around four seconds per PAP. This further confirms that our framework is capable of tackling large sets of vehicles in near real time and that it is scalable. Moreover, under these extreme conditions, we observe the largest number of unparked vehicles, namely around 70% of the vehicles were directed to their destinations with CPLEX, and 2.5% more with the greedy algorithm.

From Tables 5.I and 5.II we can conclude that CPLEX constitutes the better option, both in terms of the solution quality and execution time. However, we cannot guarantee that some vehicles will not be allocated to an unfavorable parking lot, just because there is an available slot at their arrival time. To remedy this problem, we introduce policies to the DPAP mechanism and analyze the effect they have on the solution.

Policy analysis

The policies are characterized by their parameters. In order to illustrate the impact of the parameter on the DPAP, we first analyzed its effects for a fixed time step. Namely, for 1,000 vehicles at peak traffic hours, 8 h, i.e., $k = 480$ in Lyon. We then deployed PAP(480) for various values of $\alpha(\Pi_i)$, $i = 1,2,3$ and recorded the number of unparked vehicles. The results are presented in Figure 5.7. The horizontal axis represents the value of the policy parameter α. The vertical axis represents the number of unparked vehicles. For example, we see that if the maximal walking time would be higher 25 minutes, then all vehicles would be assigned a parking lot, i.e., $P_i = P$. However, if drivers imposed a lower maximal walking time of five minutes, then around 80% vehicles would remain without a parking.

We now focus on evaluating the policies introduced in Section 4. We set four values for the parameter α following the results of Figure 5.7. More precisely, we selected values of α in the range where the slope of the curves in Figure 5.7 is most critical. For the maximal walking time policy we set the values to 10, 20, 25 and 30 minutes, i.e., the drivers will not be parked further than 10, 20, 25 or 30 minutes from their

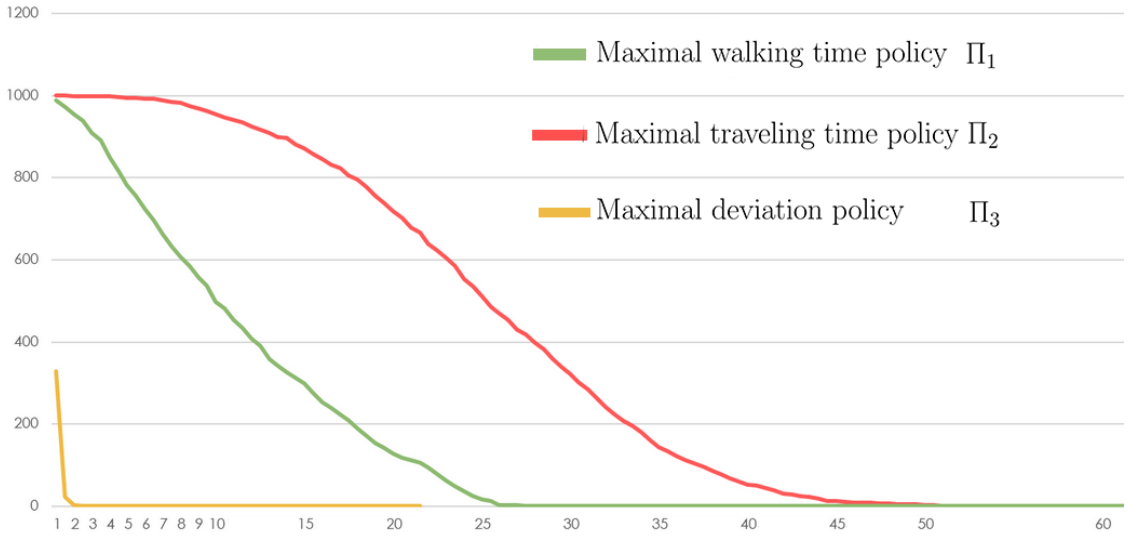

Figure 5.7 – The impact of the policy parameter α on the number of unparked vehicles.

destination. For the maximal traveling time policy, we set these values between 20 and 50 minutes. In this case the potential set of lots will be determined by the total traveling time, from the current position, and then to the destination, being less than 20, 30, 40 and 50 minutes, respectively. Finally, the value of α was set to 1.1, 1.2, 1.3 and 1.5 for the maximal deviation policy Π_3. As in Section *No-restriction policy*, two scenarios were considered: with reduced and with regular parking capacities.

Table 5.III compares the number of unparked vehicles, i.e., vehicles guided towards their destination and the total number of changes that occurred during the simulation period. The table is divided horizontally by cities and vertically by policies. The results are then reported for each value of the parameter α. The row difference in Table 5.III represent the percentage difference between the regular and reduced capacity cases. It is calculated as

$$\frac{|A-B|}{\frac{A+B}{2}} \times 100.$$

We are interested in determining which of the proposed policies will remain the least affected by the reduced parking capacities. Lower values of α cause the set P_i to be smaller, but raise the quality of individual allocations. Further, if the absolute difference between regular and reduced results are below five, we consider the gap to be non-

			Belgrade				Luxembourg				Lyon			
			$m = 23$		$\lvert V_a \rvert = 3234$		$m = 18$		$\lvert V_a \rvert = 6733$		$m = 47$		$\lvert V_a \rvert = 10683$	
			$\alpha = 10$	$\alpha = 20$	$\alpha = 25$	$\alpha = 30$	$\alpha = 10$	$\alpha = 20$	$\alpha = 25$	$\alpha = 30$	$\alpha = 10$	$\alpha = 20$	$\alpha = 25$	$\alpha = 30$
Π_1	Regular capacity	Unparked	964	30.5	7.4	2	1598.8	315.4	117.7	43.3	5250.4	916.6	91.7	17.8
		Changes	10.7	12.4	16.5	16.5	7.5	11.7	13.8	14	179.1	942	1215.7	1207.4
	Reduced capacity	Unparked	1029.6	29.9	9.4	1.6	1952	317.8	130.2	46.6	6419	1968.4	438.3	18.6
		Changes	361.6	867.3	878.8	881.8	2480.8	4980	5410.6	5626.7	5711.8	15319.4	20485.3	20988.3
	Difference	Unparked	6.6%	NS	NS	NS	19.9%	NS	10.1%	NS	20%	72.9%	130.8%	NS
		Changes	188.5%	194.4%	192.6%	192.6%	198.8%	199%	199%	199%	187.7%	176.8%	177.6%	178.2%
			$\alpha = 20$	$\alpha = 30$	$\alpha = 40$	$\alpha = 50$	$\alpha = 20$	$\alpha = 30$	$\alpha = 40$	$\alpha = 50$	$\alpha = 20$	$\alpha = 30$	$\alpha = 40$	$\alpha = 50$
Π_2	Regular capacity	Unparked	74.2	4.6	0.4	0	445.6	69.9	8.4	0.4	2050.9	73.1	7.6	1
		Changes	1448.1	355.5	30.8	17.9	343.4	79.5	23.6	15.5	5846.9	4187.6	1818.1	1236.6
	Reduced capacity	Unparked	88.4	5.7	0.3	0	476.7	76	9.3	1.2	3444.2	422.4	6.6	1
		Changes	2205.1	1255.4	898.7	886.9	5123.6	5686	5552.4	5767.6	14938.9	25808.6	23150	20052.3
	Difference	Unparked	17.5%	NS	NS	NS	6.7%	8.3%	NS	NS	50.7%	141%	NS	NS
		Changes	41.4%	111.7%	186.7%	192.1%	174.9	194.5%	198.3%	198.9%	87.5%	144.1%	170.9%	176.8%
			$\alpha = 1.1$	$\alpha = 1.2$	$\alpha = 1.3$	$\alpha = 1.5$	$\alpha = 1.1$	$\alpha = 1.2$	$\alpha = 1.3$	$\alpha = 1.5$	$\alpha = 1.1$	$\alpha = 1.2$	$\alpha = 1.3$	$\alpha = 1.5$
Π_3	Regular capacity	Unparked	83.9	68.5	51.4	34.4	173.9	146.8	120.8	84.9	722.2	494	369.9	213.8
		Changes	45.4	60.4	57.7	44.8	32.5	32.7	39.9	35.1	1380.4	1352.4	1413.8	1335.4
	Reduced capacity	Unparked	242.4	184.5	153.6	94.6	1275.8	1002.5	810.5	491.4	4586.8	3990.3	3396.3	2366.7
		Changes	786.3	800.7	865.6	855.7	4132.1	4297.6	4730.1	5025.7	8694.2	11051.3	13703.9	18577.3
	Difference	Unparked	97.1%	91.7%	99.7%	93.3%	152%	148.9%	148.1%	141.1%	145.6%	155.9%	160.7%	166.9%
		Changes	178.1%	171.9%	175%	180.1%	196.9%	197%	196.6%	197.2%	145.2%	156.4%	162.6%	173.2%

Table 5.III – Comparison of all three policies on Belgrade, Luxembourg and Lyon, for four preset values of the parameter α.

significant (NS).

Comparison per city Each of the three cities has completely different settings in terms of area, number of vehicles and number of parking lots. However, we observe several similar behaviors across the cities. For example, if the values of α are low, then up to one third of the vehicles will not be attributed a parking lot. We also recognize the pattern from the no-restriction tests, i.e., the number of allocation changes rises by up to 200% when the capacity is reduced. This provides further evidence of the responsiveness of the DPAP mechanism. As expected, the number of unparked vehicles decreases when the value of α increases. After a certain value of α, all vehicles will be assigned a parking, similarly to the case where there are no restrictions, and for similar values corresponding to Figure 5.7.

Comparison per policy From Table 5.III we recognize that Π_3 is the least able to cope with reduced capacities. The gap is never below 92% for the number of unparked vehicles. On the other hand, policies Π_1 and Π_2 converge to the same number of unparked vehicles as the value of α rises. For example, for the maximal walking time policy, we see that if the walking time is set to 30 minutes, then almost all vehicles will be assigned a parking lot. Furthermore, we observe that the total number of changes (rows

Changes in Table 5.III) increases significantly, by up to 200%, when the capacities are reduced. We also note that the reduced capacities do not influence as much the number of unparked vehicles for policies Π_1 and Π_2, where it reaches a peak of 141% for Π_2 in Luxembourg. However, this is not the case for policy Π_3, where the number of unparked vehicles rises by 167% for the city of Lyon and is never below 97%. From Table 5.III we can conclude that policies Π_1 and Π_2 are much less affected than Π_3 by the reduced capacities. Overall, the maximal traveling time policy Π_2 is more stable, yielding the least number of unparked vehicles, while the number of changes remains similar to Π_1 and Π_3. However, it also appears that if the maximal walking time is 25 minutes, then the vast majority of the vehicles will be allocated to a parking lot.

Standard deviation of all the policies Since the performed tests were based on randomly generating the vehicles on the city maps, ten runs were performed. In this paragraph we present the table of the standard deviation (STD) for the number of unparked vehicles and the number of changes for each policy for both regular and reduced capacities. The standard deviation is calculated as follows

$$\text{STD} = \sqrt{\frac{\sum_{i=1}^{10} (x_i - \bar{x})^2}{9}},$$

where x_i denotes the number of changes or the number of unparked vehicles for the run i, while $\bar{x}$ is the mean value of the ten runs.

From Table 5.IV we observe that the differences of the STD are not significantly different between the regular and reduced capacities. The biggest differences are observed in Luxembourg. This is most probably due to the fact that it represents the smallest map of the three and that the initial coordinates of the vehicles heavily influence the final result.

Table 5.V reveals similar results as Table 5.IV. The values for the regular capacity show more stable results, as expected, but we also observe that the deviation in the case of reduced capacity is relatively low.

For the maximal deviation policy, the STD values are presented in Table 5.VI. Once again we see a similar pattern of values, with more stable values for the regular capacity.

Regular capacity

	$\alpha = 20$		$\alpha = 30$		$\alpha = 40$		$\alpha = 50$	
City	Number of changes	Unparked vehicles	Number of changes	Unparked vehicles	Number of changes	Unparked vehicles	Number of changes	Unparked vehicles
Belgrade	28.71	8.40	17.16	1.50	5.73	0.52	5.95	0
Luxembourg	14.45	14.41	7.32	5.55	4.79	2.42	5.66	0.84
Lyon	110.76	41.01	64.83	7.1	78.26	2.63	89.31	0.81
Reduced Capacity								
	$\alpha = 20$		$\alpha = 30$		$\alpha = 40$		$\alpha = 50$	
Belgrade	65.53	10.29	78.54	2.0	70.37	0.48	72.93	0
Luxembourg	189.48	24.0	290.77	7.16	381.31	3.1	390.91	1.03
Lyon	118.55	35.69	169.71	22.57	372.04	2.12	295.79	0.94

Table 5.IV – Standard deviation values for the maximal traveling time policy Π_1

Regular capacity

	$\alpha = 10$		$\alpha = 20$		$\alpha = 25$		$\alpha = 30$	
City	Number of changes	Unparked vehicles	Number of changes	Unparked vehicles	Number of changes	Unparked vehicles	Number of changes	Unparked vehicles
Belgrade	3.71	32.78	4.5	3.84	5.76	2.76	5.76	1.41
Luxembourg	4.27	31.18	3.77	14.74	6.0	8.78	5.6	4.74
Lyon	33.78	39.97	79.18	28.89	82.6	8.72	91.85	4.31
Reduced Capacity								
	$\alpha = 10$		$\alpha = 20$		$\alpha = 25$		$\alpha = 30$	
Belgrade	32.9	29.05	72.02	4.38	72.32	3.24	71.6	1.17
Luxembourg	175.87	155.11	257.55	17.81	407.39	11.05	355.03	5.89
Lyon	128.92	39.76	108.0	35.15	212.49	25.74	191.20	6.17

Table 5.V – Standard deviation values for the maximal walking time policy Π_2

Still, for any values of the parameter α of the results for the reduced capacities are very similar.

Regular capacity

	$\alpha = 1.1$		$\alpha = 1.2$		$\alpha = 1.3$		$\alpha = 1.5$	
City	Number of changes	Unparked vehicles	Number of changes	Unparked vehicles	Number of changes	Unparked vehicles	Number of changes	Unparked vehicles
Belgrade	7.3	9.81	7.72	9.35	11.06	6.51	5.75	5.25
Luxembourg	7.89	9.13	7.53	8.2	6.52	10.02	5.02	8.75
Lyon	98.39	11.22	75.15	34.81	82.39	17.88	82.54	28.25
Reduced Capacity								
	$\alpha = 1.1$		$\alpha = 1.2$		$\alpha = 1.3$		$\alpha = 1.5$	
Belgrade	46.43	16.75	31.57	12.93	47.25	14.29	23.0	8.03
Luxembourg	198.57	37.11	216.64	52.95	209.15	42.86	282.56	30.43
Lyon	142.02	34.63	85.44	51.46	141.67	29.15	238.0	35.21

Table 5.VI – Standard deviation values for the maximal deviation policy Π_3

Moreover, we observe that the STDs are almost the same for all the policies, per parameter α and per city. This indicates the robustness of the DPAP framework and that consistent results are produces for each configuration.

5.3 Conclusion

We are now equipped with a good MP model that can assign parking lots to vehicles (Chapter 3), and a framework in which it can be efficiently deployed (Chapter 4). In this chapter we finally test the real-world scenario in which requests are received each minute and they are simultaneously assigned a parking lot. This sequence is a part of the DPAP framework, more precisely the online algorithm of the DPAP presented in more

details in Chapter 4. The static model is solved at each decision moment of the planning horizon. The model includes all the vehicles that have not yet reached their designated parking lot. This allowed us to reevaluate previous decisions, and depending on the vehicle flow frequency, to adapt the solution to the updated input. To validate the overall DPAP approach, we collected real parking availability data from three European cities. Furthermore, we developed a simulation environment based on these data. The wealth of the framework relies on the four layers which do not influence the complexity of MIP model, and guarantee that a parking can be offered to drivers before the next decision moment, which can be achieved within less than one minute. In addition, the number of vehicles used in our experiments was as high as 200,000, and this number can easily be increased, yet it represents the largest value found in the literature.

Our tests reveal that the greedy heuristic constitutes a good choice if the parking capacities are sufficient to accommodate all the vehicles. However, when the capacities are reduced, or the number of vehicles is large, the exact algorithm is the better choice. Tests were conducted for three policies, where the maximal traveling time policy proved to be the most stable in terms of the number of allocation changes and the number of unparked vehicles over the entire planning horizon.

CHAPTER 6

CONCLUSIONS AND PERSPECTIVES

The pursuit for a vacant parking is stressful for an individual, but on a larger scale it provokes far-reaching consequences. The chapter *Parking related problems: a general context* elaborates in more detail some of those negative effects. Furthermore, it encompasses other problems related to parking from the point of view of operations research (OR). It addresses most OR concepts and solutions for tackling them and surveys significant papers on the topic, focusing on the problem of assigning a parking to vehicles.

In this manuscript we have investigated how the parking allocation problem can be formulated, keeping in mind that we wish to solve a real-world problem, and not a hypothetical one. This led us to a dynamic problem which requires to be solved in near real time. Moreover, we observed that dynamic problems are a hot topic in the OR community and that the way they are stated mainly depends on the authors motivations and requirements. Furthermore, the parking allocation problem (PAP), while being a combinatorial and dynamic problem, is rarely treated as such in the literature. We therefore, endeavored to develop a unified framework that could incorporate the elementary prerequisites of any parking allocation.

Clearly, the PAP is a dynamic problem. Vehicles are always moving and many unpredictable events can occur. That it why most papers dealing with the problem of allocating a parking to vehicles are tackled via simulation, mainly because of the uncertainty of the events to follow and the high level of dynamism. Therefore in the chapter *Parking allocation problems: a state of the art* we investigate how researchers have dealt with other dynamic problems in transportation. This chapter demonstrated that there is no clear way to define and hence solve dynamic problems. However, we have classified PAP related papers based on the dynamic vehicle routing (DVRP) taxonomy and then proposed a classification for the PAP. We observed that the way the PAP is modeled heavily depends on the assumptions that are made. These assumptions can lead to unrealistic models, or can produce some very complex ones, which in return remove the option of providing allocations to vehicles in real-time. The second point of our classification is based on how the dynamism of the PAP is treated. We observed that most authors opt

for an online algorithm. More precisely, to sequentially solve the (static) problem as soon as new data (requests) appear. The last point of our classification focuses on the design of the combinatorial formulation of the problem that needs to be solved at a given decision moment. This formulation is usually represented by a MIP model and is used as the main decision tool in allocating parking lots to vehicles. It can be seen that the range of models varies greatly from a simple assignment problem to the very hard to solve vehicle routing problem with time windows.

The static PAP, without loss of generality, can be considered as a variant of the AP. In our study, we opted to model it as a variant of the generalized assignment problem (GAP). The GAP is NP-hard, but if some basic assumptions are made, the GAP can be made simple to exactly solve. In the case of the PAP, the assumptions include the connectivity of vehicles, geo-positioning and parking availability. All of these requirements can be easily met. For example, most cities have well dispersed parking facilities and some make them available online. Geo-positioning can be made available by almost any smartphone. The connectivity can be managed by the local authorities or by another party. These assumptions are currently all available and do not necessitate new technologies or infrastructural investments. The 0-1 model is then trivial to solve, since it possesses the integrality property. As such, it can be solved to optimality very quickly, overcoming to some extent, the high dynamic nature of the problem. For the case where the number of vehicles is large, we proposed an efficient heuristic. These results were the contribution of the chapter *The static parking allocation problem*.

The static model is not applicable in practice. More precisely, the allocations made in a previous decision moment could be completely obsolete in a near future. Therefore, the dynamic PAP is considered in the chapter *dynamic parking allocation*. In other words, we examine a mechanism to sequentially deploy the static PAP over a given time horizon. The approach we opted for is an online algorithm, which assigns parking lots to vehicles at each decision moment, as new data appear. Since we can produce an optimal solution for the static PAP quickly, we keep all the vehicles that have not yet reached their designated parking lot. This simple approach removes many potential difficulties in the dynamic setup, such as estimating stochastic variables and state transition functions. Moreover, a logical extension to the static PAP is to introduce several scenarios and policies for a more elaborate dynamic PAP case. To this end, a framework was pro-

posed, capable of tackling in near real-time these dynamic changes and incorporating the static PAP, dynamic changes, policies and real data. The framework also proved to be responsive and robust, because it is capable of providing a solution in near real time for up to 200,000 vehicles a day, and can easily surpass that number. Moreover, the results are consistent with the policies applied to the framework.

To validate our framework we have acquired real parking availability data from three European cities: Belgrade, Luxembourg and Lyon. We were also able to obtain the coordinates of the parking lots of these cities. These data allowed us to develop an environment in which we could test the DPAP framework. The vehicles originate from a random position on the city map and we assign them a random destination also within the city. The destination coordinates follow a continuous normal distribution, while the coordinates where the vehicles first appear follows a uniform distribution. We then suppose that the vehicles are moving at a constant speed and assign them parking lots at each decision moment. The decision moments were set to be one minute, mainly to test the performance of the DPAP framework and to avoid data losses.

We believe that the DPAP framework we proposed could prove to be relevant in theory and useful in practice. However, we have just scratched the surface of this rich problem. The investigation of this problem and the continuations of its refinement offers many paths. The one we have not had time to fully explore was the myopic effects of current allocations. The most common way to avoid myopic behaviour is to suppose future data and solve a future problem and then evaluate current decisions. We have opted for a data-driven approach that does not approximate future inputs, because it would take time, while our main goal was to produce a good solution as quickly as possible. Another perspective we have tested, but not sufficiently, is applying different MIP models for the static PAP. Namely, the static PAP could be a multi-modal model, in which the drivers has several options of reaching their final destination. Or the model could be multi-objective and include the prices of remaining parked for a period. This all comes down to the requirements of the classification from Section 2.5 of Chapter 2. Moreover, in this manuscript we have just slightly addressed probabilistic variables. A perspective could be to take the stochastic optimization approach to dynamic problems and test the cumulative and terminal rewards in order to compare different policies.

BIBLIOGRAPHY

[1] Abidi, Sofiene, Krichen, Saoussen, Alba, Enrique, & Bravo, Juan Miguel Molina. 2017. A Hybrid Heuristic for Solving a Parking Slot Assignment Problem for Groups of Drivers. *International Journal of Intelligent Transportation Systems Research*, **15**(2), 85–97.

[2] Amer, Ahmed, & Chow, Joseph Y.J. 2017. A downtown on-street parking model with urban truck delivery behavior. *Transportation Research Part A: Policy and Practice*, **102**, 51–67.

[3] Anderson, Simon P., & de Palma, André. 2004. The economics of pricing parking. *Journal of Urban Economics*, **55**(1), 1–20.

[4] Asmussen, Søren. 2008. *Applied probability and queues*. Vol. 51. Springer Science & Business Media.

[5] Axhausen, Kay W., & Polak, John W. 1991. Choice of parking: stated preference approach. *Transportation*, **18**, 59–81.

[6] Axhausen, Kay W., Boltze, Manfred, Polak, John W., & Puzicha, Jörg. 1994. Effectiveness of the Parking Guidance System in Frankfurt am Main. *Traffic Engineering and Control*, **35**(5), 304–309.

[7] Ayala, Daniel, Wolfson, Ouri, Xu, Bo, DasGupta, Bhaskar, & Lin, Jie. 2012. Parking in Competitive Settings: A Gravitational Approach. *Pages 27–32 of: 2012 IEEE 13th International Conference on Mobile Data Management*. IEEE.

[8] Beasley, John E. 1995. OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, **41**(11), 1069–1072.

[9] Bellman, Richard E, & Dreyfus, Stuart E. 2015. *Applied dynamic programming*. Vol. 2050. Princeton university press.

[10] Berbeglia, Gerardo, Cordeau, Jean-François, Gribkovskaia, Irina, & Laporte, Gilbert. 2007. Static pickup and delivery problems: a classification scheme and survey. *TOP*, **15**(1), 1–31.

[11] Berbeglia, Gerardo, Cordeau, Jean-François, & Laporte, Gilbert. 2010. Dynamic pickup and delivery problems. *European Journal of Operational Research*, **202**(1), 8–15.

[12] Burkard, Rainer, Dell'Amico, Mauro, & Martello, Silvano. 2009. *Assignment problems*. Philadelphia.

[13] Burkard, Rainer E., Karisch, Stefan E., & Rendl, Franz. 1997. QAPLIB – A Quadratic Assignment Problem Library. *Journal of Global Optimization*, **10**(4), 391–403.

[14] Caicedo, Felix, Lopez-Ospina, Hector, & Pablo-Malagrida, Ramon. 2016. Environmental repercussions of parking demand management strategies using a constrained logit model. *Transportation Research Part D: Transport and Environment*, **48**, 125–140.

[15] Cavadas, Joana, Homem de Almeida Correia, Gonçalo, & Gouveia, João. 2015. A MIP model for locating slow-charging stations for electric vehicles in urban areas accounting for driver tours. *Transportation Research Part E: Logistics and Transportation Review*, **75**, 188–201.

[16] Cenerario, Nicolas, Delot, Thierry, & Ilarri, Sergio. 2008. Dissemination of information in inter-vehicle ad hoc networks. *In: 2008 IEEE Intelligent Vehicles Symposium*. IEEE.

[17] Cohen, Reuven, Katzir, Liran, & Raz, Danny. 2006. An efficient approximation for the Generalized Assignment Problem. *Information Processing Letters*, **100**(4), 162–166.

[18] Cordeau, J-F, Gendreau, M., Laporte, G., Potvin, J-Y, & Semet, F. 2002. A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, **53**(5), 512–522.

[19] Cordeau, Jean-François, & Laporte, Gilbert. 2007. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, **153**(1), 29–46.

[20] Cordeau, Jean-François, Laporte, Gilbert, Legato, Pasquale, & Moccia, Luigi. 2005. Models and Tabu Search Heuristics for the Berth-Allocation Problem. *Transportation Science*, **39**(4), 526—-538.

[21] Cordeau, Jean-François, Laporte, Gilbert, Savelsbergh, Martin W.P., & Vigo, Daniele. 2007. Chapter 6 Vehicle Routing. *Pages 367–428 of:* Barnhart, Cynthia, & Laporte, Gilbert (eds), *Transportation*. Handbooks in Operations Research and Management Science, vol. 14. Elsevier.

[22] D'Acierno, Luca, Gallo, Mariano, & Montella, Bruno. 2006. Optimisation models for the urban parking pricing problem. *Transport Policy*, **13**(1), 34–48.

[23] Davis, Amélie Y., Pijanowski, Bryan C., Robinson, Kimberly D., & Kidwell, Paul B. 2010. Estimating parking lot footprints in the Upper Great Lakes Region of the USA. *Landscape and Urban Planning*, **96**(2), 68–77.

[24] Defude, Bruno, Delot, Thierry, Ilarri, Sergio, Zechinelli, Jose-Luis, & Cenerario, Nicolas. 2008. Data aggregation in VANETs: the VESPA approach. *Pages 1–13 of: Proceedings of the 5th annual international conference on mobile and ubiquitous systems: computing, networking, and services*. ICST.

[25] Delot, Thierry, Cenerario, Nicolas, Ilarri, Sergio, & Lecomte, Sylvain. 2009. A Cooperative Reservation Protocol for Parking Spaces in Vehicular Ad Hoc Networks. *Pages 30:1–30:8 of: Proceedings of the 6th International Conference on Mobile Technology, Application Systems*. Mobility '09. ACM.

[26] Delot, Thierry, Cenerario, Nicolas, & Ilarri, Sergio. 2010. Vehicular event sharing with a mobile peer-to-peer architecture. *Transportation Research Part C: Emerging Technologies*, **18**(4), 584–598.

[27] Delot, Thierry, Ilarri, Sergio, Lecomte, Sylvain, & Cenerario, Nicolas. 2013. Sharing with caution: Managing parking spaces in vehicular networks. *Mobile Information Systems*, **9**(1), 69–98.

[28] Desrosiers, Jacques, Dumas, Yvan, Solomon, Marius M, & Soumis, François. 1995. Time constrained routing and scheduling. *Handbooks in operations research and management science*, **8**, 35–139.

[29] Djenić, Aleksandar, Marić, Miroslav, Mladenović, Marko, Božović, Srđan, & Netković, Miloš. 2012. Implementation of visitor pattern in processing a syntax tree in QLab project. *Serbian Journal of Electrical Engineering*, **9**(1), 29–32.

[30] Djenić, Aleksandar, Radojičić, Nina, Marić, Miroslav, & Mladenović, Marko. 2016. Parallel VNS for Bus Terminal Location Problem. *Applied Soft Computing*, **42**, 448–458.

[31] Dsouza, Kevin Bradley, Mohammed, Syed, & Hussain, Yousuff. 2017. Smart parking—An integrated solution for an urban setting. *Pages 174–177 of: Convergence in Technology (I2CT), 2017 2nd International Conference for*. IEEE.

[32] Evenepoel, S., Van Ooteghem, J., Verbrugge, S., Colle, D., & Pickavet, M. 2014. On-street smart parking networks at a fraction of their cost: performance analysis of a sampling approach. *Transactions on Emerging Telecommunications Technologies*, **25**(1), 136–149.

[33] Farag, Mohamed S., Mohie El Din, M. M., & El Shenbary, H. A. 2017. Smart Parking Guidance Using Optimal Cost Function. *Computer and Information Science*, **10**(1), 48–53.

[34] Faria, Marta V., Baptista, Patrícia C., & Farias, Tiago L. 2014. Electric vehicle parking in European and American context: Economic, energy and environmental analysis. *Transportation Research Part A: Policy and Practice*, **64**, 110–121.

[35] Feeney, Bernard P. 1991. A review of the impact of parking policy measures on travel demand. *Transportation Planning and Technology*, **13**, 229–244.

[36] Florian, Micha el, & Los, Marc. 1980. Impact of the supply of parking spaces on parking lot choice. *Transportation Research Part B: Methodological*, **14**(1–2), 155–163.

[37] Gantelet, E., & Lefauconnier, A. 2006. The time looking for a parking space: Strategies, associated nuisances and stakes of parking management in France. Europe Transport Conference.

[38] Gendreau, Michel, Guertin, François, Potvin, Jean-Yves, & Taillard, Éric. 1999. Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching. *Transportation Science*, **33**(4), 381–390.

[39] Geng, Yanfeng, & Cassandras, C.G. 2011. Dynamic resource allocation in urban settings: A smart parking approach. *Pages 1–6 of: 2011 IEEE International Symposium on Computer-Aided Control System Design (CACSD)*.

[40] Geoffrion, Arthur. 1968. Proper Efficiency and the Theory of Vector Maximization. *Journal of Mathematical Analysis and Applications*, **22**(3), 618–630.

[41] Geoffrion, Arthur. 1974. Lagrangean relaxation for integer programming. *Pages 82–114 of: Approaches to Integer Programming*. Mathematical Programming Studies, no. 2. Springer Berlin Heidelberg.

[42] Gilbert, Kenneth C., & Hofstra, Ruth Bisgrove. 1987. An Algorithm for a Class of Three-Dimensional Assignment Problems Arising in Scheduling Applications. *IIE Transactions*, **19**(1), 29–33.

[43] Gross, O. 1959. *The bottleneck assignment problem*. Tech. rept. RAND CORP SANTA MONICA CALIF.

[44] Hanafi, Saïd, Lazić, Jasmina, Mladenović, Nenad, Wilbaut, Christophe, & Crévits, Igor. 2010. Hybrid Variable Neighbourhood Decomposition Search for 0-1 Mixed Integer Programming Problem. *Electronic Notes in Discrete Mathematics*, **36**, 883–890. ISCO 2010 - International Symposium on Combinatorial Optimization.

[45] Hanafi, Saïd, Lazić, Jasmina, Mladenović, Nenad, Wilbaut, Christophe, & Crévits, Igor. 2015. New variable neighbourhood search based 0-1 mip heuristics. *Yugoslav Journal of Operations Research*, **25**(3), 343–360.

[46] Hansen, Pierre, Mladenović, Nenad, & Moreno Pèrez, Jos e A. 2010. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, **175**(1), 367–407.

[47] Hansen, Pierre, Mladenović, Nenad, Todosijević, Raca, & Hanafi, Saïd. 2016. Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, 1–32.

[48] Hvattum, Lars M., Løkketangen, Arne, & Laporte, Gilbert. 2006. Solving a Dynamic and Stochastic Vehicle Routing Problem with a Sample Scenario Hedging Heuristic. *Transportation Science*, **40**(4), 421–438.

[49] Hvattum, Lars Magnus, Løkketangen, Arne, & Laporte, Gilbert. 2007. A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. *Networks*, **49**(4), 330–340.

[50] Ioakimidis, Christos S., Thomas, Dimitrios, Rycerski, Pawel, & Genikomsakis, Konstantinos N. 2018. Peak shaving and valley filling of power consumption profile in non-residential buildings using an electric vehicle parking lot. *Energy*, **148**, 148–158.

[51] Isermann, H. 1974. Technical Note—Proper Efficiency and the Linear Vector Maximum Problem. *Operations Research*, **22**(1), 189–191.

[52] Jannati, Jamil, & Nazarpour, Daryoosh. 2019. Optimal performance of electric vehicles parking lot considering environmental issue. *Journal of Cleaner Production*, **206**, 1073–1088.

[53] Kogan, Konstantin, & Shtub, Avraham. 1997. DGAP - The Dynamic Generalized Assignment Problem. *Annals of Operations Research*, **69**, 227–239.

[54] Kuhn, H. W. 2005. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, **52**(1), 7–21.

[55] Laguna, Manuel, Kelly, James P., González-Velarde, JoséLuis, & Glover, Fred. 1995. Tabu search for the multilevel generalized assignment problem. *European Journal of Operational Research*, **82**(1), 176–189.

[56] Laporte, G., Gendreau, M., Potvin, J-Y., & Semet, F. 2000. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, **7**(4-5), 285–300.

[57] Laporte, Gilbert. 1992. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, **59**(3), 345–358.

[58] Laporte, Gilbert. 2009. Fifty Years of Vehicle Routing. *Transportation Science*, **43**(4), 408–416.

[59] Laporte, Gilbert, & Louveaux, François V. 1993. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, **13**(3), 133–142.

[60] Laporte, Gilbert, & Osman, Ibrahim H. 1995. Routing problems: A bibliography. *Annals of Operations Research*, **61**(1), 227–262.

[61] Lin, Trista, Rivano, Herve, & Le Mouel, Frederic. 2017. A Survey of Smart Parking Solutions. *IEEE Transactions on Intelligent Transportation Systems*, **18**(12), 3229–3253.

[62] Louveaux, François V., & van der Vlerk, Maarten H. 1993. Stochastic programming with simple integer recourse. *Mathematical Programming*, **61**(1), 301–325.

[63] Mackowski, Daniel, Bai, Yun, & Ouyang, Yanfeng. 2015. Parking Space Management via Dynamic Performance-based Pricing. *Transportation Research Procedia*, **7**, 170–191. 21st International Symposium on Transportation and Traffic Theory Kobe, Japan, 5-7 August, 2015.

[64] McShane, Mary, & Meyer, Michael D. 1982. Parking policy and urban goals: Linking strategy to needs. *Transportation*, **11**(2), 131–152.

[65] Millard-Ball, Adam. 2019. The autonomous vehicle parking problem. *Transport Policy*, **75**, 99–108.

[66] Millard-Ball, Adam, Weinberger, Rachel R., & Hampshire, Robert C. 2014. Is the curb 80% full or 20% empty? Assessing the impacts of San Francisco's parking pricing experiment. *Transportation Research Part A: Policy and Practice*, **63**, 76–92.

[67] Mitrović-Minić, Snežana, & Punnen, Abrahim. 2008a. Very large-scale variable neighborhood search for the generalized assignment problem. *Journal of Interdisciplinary Mathematics*, **11**(5), 653–670.

[68] Mitrović-Minić, Snežana, & Punnen, Abrahim P. 2008b. Very large-scale variable neighborhood search for the generalized assignment problem. *Journal of Interdisciplinary Mathematics*, **11**(5), 653–670.

[69] Mitrović-Minić, Snežana, Krishnamurti, Ramesh, & Laporte, Gilbert. 2004. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, **38**(8), 669–685.

[70] Mjirda, Anis, Jarboui, Bassem, Mladenović, Jelena, Wilbaut, Christophe, & Hanafi, Saïd. 2014. A general variable neighbourhood search for the multi-product inventory routing problem. *IMA Journal of Management Mathematics*, **27**(1), 39–54.

[71] Mladenović, Marko, Delot, Thierry, Laporte, Gilbert, & Wilbaut, Christophe. 2016 (October). Time dependent parking lot allocation problem. *In: 4rd international conference on VNS*.

[72] Mladenović, Marko, Delot, Thierry, Laporte, Gilbert, & Wilbaut, Christophe. 2017 (February). Variable Neighbourhood Decent for Parking Allocation Problem. *In: ROADEF 2017*.

[73] Mladenović, Marko, Delot, Thierry, Laporte, Gilbert, & Wilbaut, Christophe. 2018a (July). Dynamic parking allocation problem in urban areas. *In: EURO 2018*.

[74] Mladenović, Marko, Hanafi, Saïd, Delot, Thierry, Laporte, Gilbert, & Wilbaut, Christophe. 2018b (May). The extended parking allocation model for connected vehicles. *In: BALCOR 2018*.

[75] Mladenović, Marko, Delot, Thierry, Laporte, Gilbert, & Wilbaut, Christophe. 2018c. Parking Allocation Problem for connected vehicles. *Journal of Heuristics*, 1–23.

[76] Mladenović, Marko, Delot, Thierry, Laporte, Gilbert, & Wilbaut, Christophe. 2019a (May). On-line car parking via 0-1 programming. *In: Optimization Days 2019*.

[77] Mladenović, Marko, Delot, Thierry, Gilbert, Laporte, & Wilbaut, Christophe. 2019b. A scalable dynamic parking allocation framework. *Computers & Operations Research*, **submitted**.

[78] Mladenovic, Nenad. 1995. A variable neighborhood algorithm-a new metaheuristic for combinatorial optimization. *In: papers presented at Optimization Days*, vol. 12.

[79] Mladenović, Nenad, & Hansen, Pierre. 1997. Variable neighborhood search. *Computers & Operations Research*, **24**(11), 1097–1100.

[80] Moccia, Luigi, Cordeau, Jean-François, Monaco, Maria Flavia, & Sammarra, Marcello. 2009. A column generation heuristic for a dynamic generalized assignment problem. *Computers & Operations Research*, **36**(9), 2670–2681.

[81] Noel, Errol C. 1988. Park-and-Ride: Alive, Well, and Expanding in the United States. *Journal of Urban Planning and Development*, **114**(1), 1–13.

[82] Nunes, Pedro, Figueiredo, Raquel, & Brito, Miguel C. 2016. The use of parking lots to solar-charge electric vehicles. *Renewable and Sustainable Energy Reviews*, **66**, 679–693.

[83] Osman, Ibrahim H. 1995. Heuristics for the Generalised Assignment Problem: Simulated Annealing and Tabu Search Approaches. *OR Spektrum*, **169**, 211–225.

[84] Pentico, David W. 2007. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, **176**(2), 774–793.

[85] Piccioni, Cristiana, Valtorta, Marco, & Musso, Antonio. 2019. Investigating effectiveness of on-street parking pricing schemes in urban areas: An empirical study in Rome. *Transport Policy*, **80**, 136–147.

[86] Pillac, Victor, Gendreau, Michel, Guéret, Christelle, & Medaglia, Andrés L. 2013. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, **225**(1), 1–11.

[87] Powell, Warren B. 2019. A unified framework for stochastic optimization. *European Journal of Operational Research*, **275**(3), 795–821.

[88] Psaraftis, Harilaos N. 1980. A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem. *Transportation Science*, **14**(2), 130–154.

[89] Psaraftis, Harilaos N., Wen, Min, & Kontovas, Christos A. 2016. Dynamic vehicle routing problems: Three decades and counting. *Networks*, **67**(1), 3–31.

[90] Ratli, Mustapha. 2014. *Parking management system in a dynamic and multi-objective environment*. Ph.D. thesis, Université de Valenciennes et du Hainaut-Cambrésis.

[91] Roca-Riu, Mireia, Fernández, Elena, & Estrada, Miquel. 2015. Parking slot assignment for urban distribution: Models and formulations. *Omega*, **57, Part B**, 157–175.

[92] Rousseeuw, Peter J. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, **20**, 53–65.

[93] Savelsbergh, Martin. 1997. A Branch-and-Price Algorithm for the Generalized Assignment Problem. *Operations Research*, **45**(6), 831–841.

[94] Schrijver, Alexander. 1986. *Theory of Linear and Integer Programing*. Wiley & Sons.

[95] Shao, Chaoyi, Yang, Hai, Zhang, Yi, & Ke, Jintao. 2016. A simple reservation and allocation model of shared parking lots. *Transportation Research Part C: Emerging Technologies*, **71**, 303–312.

[96] Shmoys, David B., & Tardos, Éva. 1993. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, **62**(1), 461—-474.

[97] Shoup, Donald. 2011. *The High Cost of Free Parking*. Chicago: Planners Press.

[98] Shoup, Donald C. 1997. High cost of free parking. *Journal of Planning Education and Research*, **17**, 3–22.

[99] Shoup, Donald C. 2006. Cruising for parking. *Transport Policy*, **13**(6), 479–486.

[100] Spivey, Michael Z., & Powell, Warren B. 2004. The Dynamic Assignment Problem. *Transportation Science*, **38**(4), 399–419.

[101] Taillard, Éric. 1991. Robust taboo search for the quadratic assignment problem. *Parallel computing*, **17**(4-5), 443–455.

[102] Tang, Chaogang, Wei, Xianglin, Zhu, Chunsheng, Chen, Wei, & Rodrigues, Joel J. P. C. 2018. Towards Smart Parking Based on Fog Computing. *IEEE Access*, **6**, 70172–70185.

[103] Teodorović, Dušan, & Lučić, Panta. 2006. Intelligent parking systems. *European Journal of Operational Research*, **175**(3), 1666–1681.

[104] Thompson, Russell G, & Bonsall, Peter. 1997. Drivers' response to parking guidance and information systems. *Transport Reviews*, **17**(2), 89–104.

[105] Todosijević, Raca, Mladenović, Marko, Hanafi, Saïd, & Crévits, Igor. 2012. VNS based heuristic for solving the Unit Commitment problem. *Electronic Notes in Discrete Mathematics*, **39**, 153–160.

[106] Todosijević, Raca, Mladenović, Marko, Hanafi, Saïd, Mladenović, Nenad, & Crévits, Igor. 2016. Adaptive general variable neighborhood search heuristics for solving the unit commitment problem. *International Journal of Electrical Power & Energy Systems*, **78**, 873–883.

[107] Todosijević, Raca, Mjirda, Anis, Mladenović, Marko, Hanafi, Saïd, & Gendron, Bernard. 2017. A general variable neighborhood search variants for the travelling salesman problem with draft limits. *Optimization Letters*, **11**(6), 1047–1056.

[108] Toutouh, Jamal, & Alba, Enrique. 2016. Distributed Fair Rate Congestion Control for Vehicular Networks. *Pages 433–442 of: Distributed Computing and Artificial Intelligence, 13th International Conference*. Advances in Intelligent Systems and Computing, no. 474. Springer International Publishing.

[109] Ulungu, E. L., & Teghem, J. 1994. Multi-objective combinatorial optimization problems: A survey. *Journal of Multi-Criteria Decision Analysis*, **3**(2), 83–104.

[110] Verroios, Vasilis, Efstathiou, Vasilis, & Delis, Alex. 2011. Reaching available public parking spaces in urban environments using ad hoc networking. *Pages 141–151 of: Mobile Data Management (MDM), 2011 12th IEEE International Conference*, vol. 1. IEEE.

[111] Wen, Min, Cordeau, Jean-François, Laporte, Gilbert, & Larsen, Jesper. 2010. The dynamic multi-period vehicle routing problem. *Computers & Operations Research*, **37**(9), 1615 – 1623.

[112] White, Douglas J. 1984. A special multi-objective assignment problem. *Journal of the Operational Research Society*, **35**(8), 759–767.

[113] Xu, Bo, Wolfson, Ouri, Yang, Jie, Stenneth, Leon, Philip, S Yu, & Nelson, Peter C. 2013. Real-time street parking availability estimation. *Pages 16–25 of: 2013 IEEE 14th International Conference on Mobile Data Management*, vol. 1. IEEE.

[114] Yagiura, Mutsunori, Ibaraki, Toshihide, & Glover, Fred. 2006. A path relinking approach with ejection chains for the generalized assignment problem. *European Journal of Operational Research*, **169**(2), 548–569.

[115] Yang, Wenjing, & Lam, Patrick T.I. 2019. Evaluation of drivers' benefits accruing from an intelligent parking information system. *Journal of Cleaner Production*, **231**, 783–793.

[116] Yokomi, Muneki, Wheat, Phill, & Mizutani, Jun. 2017. The impact of low cost carriers on non-aeronautical revenues in airport: An empirical study of UK airports. *Journal of Air Transport Management*, **64**, 77–85.

[117] Zheng, Nan, & Geroliminis, Nikolas. 2016. Modeling and optimization of multimodal urban networks with limited parking and dynamic pricing. *Transportation Research Part B: Methodological*, **83**, 36–58.

[118] Zou, Bo, Kafle, Nabin, Wolfson, Ouri, & Lin, Jie (Jane). 2015. A mechanism design based approach to solving parking slot assignment in the information era. *Transportation Research Part B: Methodological*, **81**, 631–653.

LIST OF FIGURES

LIST OF TABLES