A scalable dynamic parking allocation framework

Marko Mladenovic, Thierry Delot, Gilbert Laporte, Christophe Wilbaut

Elsevier Editorial System(tm) for Computers & Operations Research

Manuscript Draft

Manuscript Number: COR-D-19-00991R3

Title: A scalable dynamic parking allocation framework

Article Type: Research Article

Keywords: Parking allocation; Dynamism; Assignment problems; 0-1 programming

Corresponding Author: Dr. Marko Mladenovic,

Corresponding Author's Institution:

First Author: Marko Mladenović

Order of Authors: Marko Mladenović; Thierry Delot; Gilbert Laporte; Christophe Wilbaut

Abstract: Cities suffer from high traffic congestion of which one of the main causes is the unorganized pursuit for available parking. Apart from traffic congestion, the blind search for a parking slot causes financial and environmental losses. We consider a general parking allocation scenario in which the GPS data of a set of vehicles, such as the current locations and destinations of the vehicles, are available to a central agency which will guide the vehicles toward a designated parking lot, instead of the entered destination. In its natural form, the parking allocation problem is dynamic,
i.e., its input is continuously updated. Therefore, standard static allocation and assignment rules do not apply in this case. In this paper, we propose a framework capable of tackling these real-time updates. From a methodological point of view, solving the dynamic version of the parking allocation problem represents a quantum leap compared with solving the static version. We achieve this goal by solving a sequence of 0-1 programming models over the planning horizon, and we develop several parking policies. The proposed policies are empirically compared on real data gathered from three European cities: Belgrade, Luxembourg, and Lyon. The results show that our framework is scalable and can improve the quality of the allocation, in particular when parking capacities are low.

Cover Letter

Dear editor,

Please find enclosed the revised version of our paper entitled: A scalable dynamic parking allocation framework. In it, we address the topic of dynamically assigning parking lots to vehicles over a given time horizon. We formulate the problem via a 0-1 programming model, which is sequentially solved over time. By applying several simple principles, we are able to propose a dynamical formulation of the problem, which assigns parking lots to vehicles depending on the current traffic updates. The results indicate that it can process a significant number of vehicles and that, in cases of reduced parking capacities, it offers alternatives to vehicles that would previously remain unparked.

The framework we developed is highly scalable and responsive, capable of assigning parking lots to vehicles in near real-time. This is demonstrated on tests conducted on real data collected from three European cities.

We believe that this study carries interesting ideas which could be applied on other problems and extended. Furthermore, it represents a clear-cut approach to solve a real-world problem for which the attention will only grow in the following years and for which there is still no consistent formulation in the literature. Therefore, we hope that you share our motivation for this interesting research topic and recognize the results of our study as fitting for one of the future issues of your journal.

Sincerely,

The authors

Minor revisions to the paper entitled
A scalable dynamic parking allocation framework
by Marko Mladenovic, Thierry Delot, Gilbert Laporte, Christophe Wilbaut

Editor

Please make the small adjustments suggested by reviewer 3.

The revisions pointed out by the Reviewer 3 were corrected.

1 Reviewer 1

Given the last developments the authors have added to the paper, I do think that the paper in its present form deserve to be published in Computers and Operations. I think the model and the ideas discussed in the paper are very interesting and I am looking forward to see further developments.

Once again, we thank you for your valuable comments and interesting remarks. We would also like to extend our gratitude for your compliments to our work.

2 Reviewer 3

The authors have satisfactory dealt with my major remarks. I only have a few minor comments.

We are grateful for your insightful comments and remarks. They have truly helped us improve our paper. The attention to details was most kind. Please find our answers to your minor revisions highlighted in the blue color in the manuscript.

- *Page 14 line 275: replace 'how would a heuristic cope' by 'how a heuristic would cope'.*
  DONE.
- *Algorithm 2 line 3: add some more space between T'' and t^* since now it seems to be a product while it is an explanation between brackets.*
  DONE.
- *Algorithm 2 line 4: remove brackets around T^*.*
  DONE.

- *Algorithm 2 line 8: shouldn't k be t_k?*
  DONE.
- *Page 14 line 283: add space between 'times.' and 'The'.*
  DONE.
- *Page 33 line 601: rewrite 'The dynamic PAP framework demonstrated it could provide' to, for example, 'We have demonstrated that the dynamic PAP framework could provide'.*
  DONE.
- *Page 33 line 606: replace 'effect current decision' by 'effect of the current decision'.*
  DONE.

Highlights of the revised paper

1. New tests were conducted to evaluate the fairness of individual allocations
2. The explanations surrounding the greedy heuristic was improved
3. Section 5.3 and the congestion effects were rewritten
4. Misprints and other imprecisions were corrected

A scalable dynamic parking allocation framework

Marko Mladenović[a,*], Thierry Delot[a], Gilbert Laporte[b,c], Christophe Wilbaut[a]

[a] *UPHF, LAMIH UMR CNRS 8201, Mont Houy, 59313 Valenciennes, France*
[b] *HEC Montréal, 3000, chemin de la Côte-Sainte-Catherine, H3T 2A7 Montréal, Canada*
[c] *School of Management, University of Bath, Bath, United Kingdom*

Abstract

Cities suffer from high traffic congestion of which one of the main causes is the unorganized pursuit for available parking. Apart from traffic congestion, the blind search for a parking slot causes financial and environmental losses. We consider a general parking allocation scenario in which the GPS data of a set of vehicles, such as the current locations and destinations of the vehicles, are available to a central agency which will guide the vehicles toward a designated parking lot, instead of the entered destination. In its natural form, the parking allocation problem is dynamic, i.e., its input is continuously updated. Therefore, standard static allocation and assignment rules do not apply in this case. In this paper, we propose a framework capable of tackling these real-time updates. From a methodological point of view, solving the dynamic version of the parking allocation problem represents a quantum leap compared with solving the static version. We achieve this goal by solving a sequence of 0-1 programming models over the planning horizon, and we develop several parking policies. The proposed policies are empirically compared on real data gathered from three European cities: Belgrade, Luxembourg, and Lyon. The results show that our framework is scalable and can improve the quality of the allocation, in particular when parking capacities are low.

Keywords: `Parking allocation, Dynamism, Assignment problems, 0-1 programming`

1. Introduction

Several studies indicate that a significant percentage of traffic congestion in urban areas is due to vehicles searching for vacant parking slots [1, 2, 3, 4]. The time spent to find a vacant slot causes

*Corresponding author `marko.mladenovic@uphf.fr`
Email addresses: `marko.mladenovic@uphf.fr` (Marko Mladenović), `thierry.delot@uphf.fr` (Thierry Delot), `gilbert.laporte@cirrelt.net` (Gilbert Laporte), `christophe.wilbaut@uphf.fr` (Christophe Wilbaut)

a number of adverse outcomes, including CO_2 emissions and wasted time. For example, in 2001 it was estimated that at least 730 tons of CO_2 were emitted from cars searching for available parking in a small Los Angeles business district [5]. In terms of wasted time, a study has estimated that on average, a driver in the UK will spend 106 days of his life just looking for parking[1]. In France a similar effect has been reported in [3].

One recurring conclusion of several studies is that there are usually sufficient parking slots to accommodate all the vehicles, and the construction of new parking lots is not recommended [6]. Furthermore, it is perceived that curb parking is not necessarily the best option, even though it may seem the cheapest one [1, 2], hence the use of parking lots is recommended. Therefore a natural goal is to allocate vehicles to existing facilities in order to minimize some of the adverse effects associated with the quest for parking.

Allocating parking facilities to vehicles has been studied for more than 40 years, and one of the first surveys appeared in [7]. Although the problem of allocating parking facilities to cars is clearly combinatorial, there is no uniform mathematical programming (MP) model for it. We address this uncharted field and introduce a framework capable of coping with real-time updates. Solving the dynamic version of the parking allocation problem represents a quantum leap compared with solving the static version. First, in the dynamic case, it is necessary to solve a vast number of static problems as opposed to only one. Second, the solutions must be computed in real time and the associated algorithm must be very fast. We have therefore developed a very efficient solution methodology that leverages our previous work on the solution of the static problem in order to make it applicable to the dynamic case. This is achieved by solving a sequence of 0-1 models over the planning horizon, under several parking policies. In this study, we ignore the effect of parking pricing mechanisms which are a recurring theme in transportation economics. We assume that the drivers who wish to take part in the parking allocation system pay a yearly fee which they judge to be fair from a market standpoint. In other words, we do not wish to determine what this fee should be. Besides, we do not focus on traffic management itself as in [8] or on a guidance system as in [9] and more recently [10]. We first briefly survey the literature on papers which take a combinatorial standpoint, static or dynamic, of the parking allocation problem (PAP).

[1] http://www.telegraph.co.uk/motoring/news/10082461/Motorists-spend-106-days-looking-for-parking-spots.html

1.1. Related work

Optimally assigning parking lots to a set of vehicles can be dealt with in various ways. The problem can be considered as a variant of the assignment problem, such as in [11]. 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. Their model possesses properties similar to the one considered in [12]. In the latter paper, the authors consider a set of interconnected vehicles with capacity and allocation constraints. They prove that their model possesses the integrality property and propose a heuristic to allocate parking lots to up to 90,000 vehicles. A similar model was proposed in [13], which considered the shared use of residential parking spaces between residents and public users. More recently, an assignment problem-based model was published [14], 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. Several greedy algorithms were developed for the proposed model and simulations were conducted to evaluate their efficiency over a section of Xuzhou City, covering six parking lots and up to 650 parking requests. All of the previously mentioned models are linear boolean MP models.

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, as suggested in [15]. The authors of this paper consider the limited parking availability for distribution vehicles and the strict timetable they have to respect. The search space is bounded by linear constraints while several potential objective functions, both linear and non-linear, are considered. These different objective functions are used to introduce several mixed integer formulations that are then solved by a standard MP solver.

The problem can also be stated as a variant of the traveling salesman problem (TSP), called the time-varying TSP (TVTSP), as in [16]. The TVTSP is also a boolean linear model. 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 they group the vehicles into clusters to improve the algorithm's efficiency.

From what can be observed today, it appears that electric and autonomous vehicles (EV and AV, respectively) will replace the petrol-fueled cars in a not so distant future. A simple search on Science Direct with the key words: "electric vehicle" and "parking" resulted in more than

3,500 papers published since 2015 and 1,633 results for the period between 2010 and 2015. This significant rise of 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 ([17]).

The main disadvantage of EVs is the same since their creation in the 19th century, i.e., their limited range ([18]). Moreover, their charging times are also significant when compared with combustion vehicles. This is one of the main reasons why their parking is important: they can be recharged while parked.

Autonomous vehicles have a particularly interesting feature: they do not need to be parked close to their destination, or even be parked at all. In [19], an autonomous vehicle parking problem is formulated that simulates a game theoretic model based on data from San Francisco. The author 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 number of trips made in urban areas.

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 formulations in the field of transportation grown rapidly [20, 21, 22]. 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 with parking assignment. However, most papers dealing with vehicle parking that have appeared in recent years fall under the umbrella of smart cities [4, 23, 24], and focus on information collection, system deployment and service dissemination, see [4, 25, 26], for instance.

1.2. A new approach

In this paper, we consider parking allocation as a combinatorial problem which is continuously updated. As in [11, 12, 13, 14], we have opted to consider the combinatorial part of the problem as a variant of the assignment problem. Unlike [13, 14, 16], we expect no other input information from the users apart from their destination. Furthermore, as in most articles [11, 14, 15, 16, 27, 25, 28, 29], we consider the vehicles to be interconnected within a network and we assume that the data are available to a central server. In particular, the number of available parking spaces at any given

time is known to the server, which means that we do not have to consider exit patterns from the parking facilities. Theses assumptions are nowadays completely realistic with the development of information systems and communication tools. An important point is that in contrast to these papers, we do not assume that we control all the vehicles in the traffic, which means that we must rely on real parking availability data and cannot impose flow constraints. We assume that whenever several parking slots are available within the same parking lot, each car is assigned to its most convenient slot. Moreover, since we do not have data about all the vehicles we do not assume full user compliance, but only on those who rely on our guidance system. In other words, users will not be competing against each other for a better parking slot. In fact, even if some drivers using our system do not follow the instructions, or even if drivers not using our system occupy a place assigned by our system, the updated number of available places will be taken into account in the next decision moment. This allows our framework to look for an updated solution.

This paper aims to present a new mechanism which could be used to coordinate the parking allocation of vehicles on a large scale. To this end, we propose a flexible and scalable framework capable of tackling the various dynamic changes in the problem at hand. We rely on the recent static PAP model proposed in [12], since it incorporates the constraints associated with the dynamic PAP. Note that our framework is not a reservation system that books parking slots in advance.

The performance of our approach can be measured by the number of vehicles and parking lots that can be coordinated in near real time, which is larger than in any other paper that we could find [11, 14, 15, 27, 29]. A common cause of traffic congestion is that most vehicles are heading to the same location. Since in peak hours most vehicles drive to just a couple of locations, assigning vehicles to parking lots, instead of their destination, can avoid bottleneck effects in the streets [27, 30]. To this end, we also take into consideration that the vehicle speed will be reduced in peak hours as proposed in [31].

1.3. Scientific contribution and outline

The scientific contributions of this paper can be summarized as follows: (i) we propose and develop a four-layer scalable framework capable of easily assigning parking lots to a large set of vehicles in near real time; (ii) we develop a mechanism that coordinates a sequence of static PAPs, called dynamic PAP, following the strategic rules of a given policy; (iii) to evaluate our framework we develop a simulated environment based on real data that we have collected from three European

cities: Belgrade, Luxembourg and Lyon.

The remainder of this paper is organized as follows. Section 2 introduces the 0-1 programming model which corresponds to the dynamic PAP. Section 3 presents the configuration coordinating the dynamic PAP sequence. Section 4 sets up the environment for the simulation. Computational experiments on real data are detailed in Section 5, while Section 6 concludes the paper.

2. The Dynamic Parking Allocation Problem

As mentioned in the previous section, the dynamic formulation of the parking allocation problem has been addressed in multiple ways. However, the available studies either state a model as dynamic and do not incorporate input updates, or are limited by the volume of vehicle requests when dealing with input updates, e.g., [14]. In the following section we introduce our framework, which successfully tackles both aspects.

2.1. General framework

The framework we propose is structured in four layers: (i) the policies which determine the dynamic setup, (ii) the dynamic PAP, (iii) the mixed integer programming (MIP) model for solving the PAP, and (iv) the collected data.

The policies determine the decision moments and the subset of parking lots to which the dynamic PAP (DPAP) will be applied. The DPAP is the mechanism that handles the dynamic nature of the problem, the continuous updating of vehicles and their large number in real time. The static PAP is the tool that allocates parking lots to vehicles. It is defined by means of a 0-1 programming model and 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.

This framework is scalable in regard to several criteria: the number of vehicles that can be handled in near real time, the static PAP layer which can be any combinatorial optimization model, and the fact that most changes to the framework would not increase its complexity and would yield real-time results.

	Notation	Size	Time-dependent	Definition
Sets	H	$\|H\| = h$	✗	Planning horizon
	V^k	depends on k	✓	Set of active 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 lots of vehicle i
	A	$\|A\| \leq \sum_{l=1}^{k} \|V^l\|$	✓	Set of vehicles arrived to their parking lots at time k
	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 to destination i
	τ_i	/	✓	Time vehicle i spends in the system
	q_j	/	✗	Total capacity of parking j
Simulation	S	/	✗	Approximation of the city center
	R	/	✗	City map
	(lat, lon)	/	✓	Geographical coordinates in latitude, longitude format
	γ	/	✗	Vehicle multiplier
	n^k_{new}	/	✓	Number of vehicles appearing at time step k

Table 1: Notation summary

2.2. 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 active 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 cardinality 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 lot $j \in P$ 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} ([12]). 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 1).

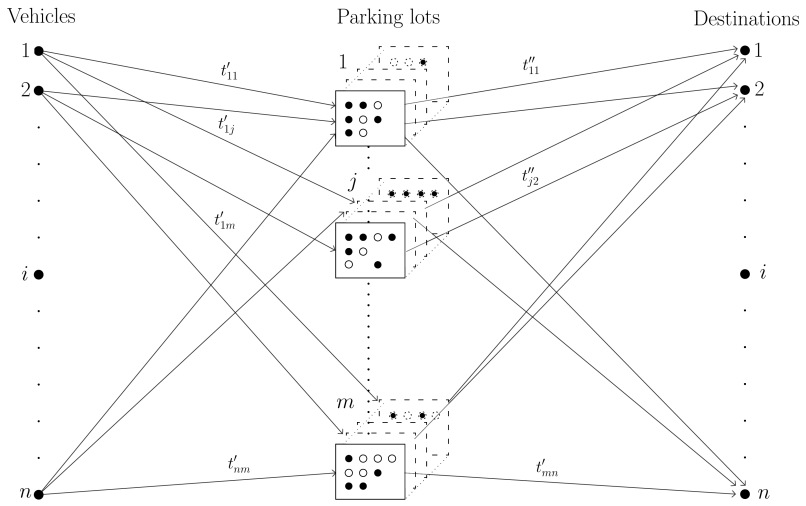

Figure 1: Representation of the DPAP for one decision moment.

The left-hand side circles of Figure 1 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 H.

The set of vehicles that have arrived at their assigned parking is denoted by A. The number of time steps a vehicle i has spent in DPAP 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.

As mentioned in Section 2.1, 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 set of decision moments $\Delta \subseteq H$ at which the DPAP will be deployed. To demonstrate the responsiveness and flexibility of our framework we set each time step to be a decision moment, i.e. $\Delta = \{\Delta_1, \ldots, \Delta_h\}$ such that $\Delta_k = t_k, \forall k = 1, \ldots, h$. The notation is summarized in Table 1.

2.3. 0-1 programming model PAP(k)

The basis of our mathematical programming model of the PAP has already been proposed in [12]. For completeness and clarity, we summarize its main components and add modifications that depend on t_k.

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

$$\text{minimize} \sum_{i \in V^k} \sum_{j \in P} \left[t'_{ij} + t''_{ji} \right] x_{ij} \tag{1}$$

$$\sum_{j \in P_i} x_{ij} = 1 \qquad i \in V^k \tag{2}$$

$$\sum_{i \in V^k} \alpha^t_{ij} x_{ij} \leq c_{jt} \qquad j \in P, t \in H \tag{3}$$

$$x_{ij} \in \{0, 1\} \qquad i \in V^k, j \in P, \tag{4}$$

where

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

The objective (1) 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 (2) 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 (3) mean that the number of allocated vehicles to a parking j will not exceed the current capacity c_{jt} at the arrival time $t'_{ij} + t_k$.

The model (1)–(4) possesses the integrality property. This implies that a linear programming solver can be used to solve it to optimality in near real time.

Residual capacity. The static model relies on the residual capacity to guarantee available slots to vehicles upon their arrival. However, if a vehicle is allocated to parking j and its arrival time is t, we cannot update the residual capacity. This is because the value of $C = (c_{jt})$ takes into account all the vehicles in traffic and an external vehicle could potentially occupy a slot that was previously allocated to a vehicle using our allocation system.

Dummy parking lot. If the number of vehicles exceeds the capacity (constraints (3)), there will be no feasible solution to the model (1)–(4). Therefore, we include a dummy parking lot $m+1$, $P = P \cup \{m+1\}$, with a large residual capacity. 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. 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 from 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.

The sequence PAP(k), $k = 1, \ldots, h$, represents the foundation of the DPAP. The objective function (1) cannot be used to attribute a value to the DPAP. Hence, to quantify the value of the DPAP, the objective function is calculated as follows:

$$f^{\text{DPAP}} = \sum_{i \in A} \left[\tau_i + t''_{j(i),i}\right], \tag{5}$$

where $j(i)$ is the parking assigned to vehicle i and A is the set of vehicles that have arrived at their assigned parking.

3. 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 H (e.g., 24 hours) discretized into h time steps, allowing reallocating vehicles in a future decision moment; (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 solve the static PAP we implement both a heuristic and an exact algorithm. We can expect that a simple greedy algorithm can provide good results when the parking capacities are high

enough. However, since we deploy the static PAP over the entire planning horizon and allow vehicle reallocations, it is less clear how a heuristic would behave overall. More precisely, how would the gap between static PAPs evolve over time compared to the exact solutions. To allow reallocations and to avoid conflicting solutions over different decision moments $\Delta_k, k = 1, \ldots, h$, all vehicles not yet parked remain 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 slot determined after the first request. From a practical point of view it is important to notice that a user of the system is not necessarily notified at once. The system can keep the assignment and notify the user when the "final" assignment is known.

3.1. 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 at time t_k will be

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

where V_a^k and V_{new}^k represent the vehicles that have arrived at their parking at time k and vehicles appearing at time k, respectively. We call V^k the *set of active vehicles* and denote its cardinality by n^k and by n_{new}^k the cardinality of V_{new}^k. Note that $\bigcup_{k \in H} V_a^k = A$.

In this paper, we do not assume that all the vehicles in traffic participate in our system, i.e., we consider an exogenous system. This further emphasizes the difficulty of conflicting solutions, because a vehicle outside of our system could occupy a slot. To guarantee an available slot, independently from the percentage of vehicles participating in our system, we make use of the residual capacity $C = (c_{jt}), j = 1, \ldots, m, t = 1, \ldots, h$. More precisely, the residual capacity is based on real data collected over time, or in real-time, from the *real data* layer of the DPAP framework. It includes both vehicles participating in the DPAP and those that do not. Moreover, this fact implies that we cannot impose flow constraints because we just know the number of vehicles at a parking lot, but not the exact number that left or arrived.

By keeping all the vehicles still seeking a parking in the set of active vehicles, this allows us (i) to wait and allocate them a lot in a future time step if no such allocation can be made at current 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.

3.2. Dynamic PAP algorithm

The previous section introduced the 0-1 MP model which will be used in the dynamic setup. The steps of our DPAP are presented in Algorithm 1.

Algorithm 1 Dynamic parking allocation routine

Function DPAP (P, C, h)
  Initialization
  $A \leftarrow \emptyset$; ▷ The list of arrived vehicles is initialized
  $V^1 \leftarrow \{1, ..., n^1_{new}\}$; ▷ The set of first n vehicles
  Compute $T'(V^1), T''(V^1)$; ▷ Compute driving and walking times
  $x_1 \leftarrow \text{PAP}(n^1_{new}, P, T', T'', C)$; ▷ Get the initial solution x_1
  Simulation loop
  for $k \leftarrow 2$ *to* h **do** ▷ For all decision moments k
    Update(V^{k-1}, x_{k-1}); ▷ Updating vehicles positions accordingly
    $A \leftarrow A \cup V^k_a$; ▷ Add vehicles arrived at time k
    $V^k \leftarrow V^k_{new} \cup (V^{k-1} \setminus V^k_a)$; ▷ Add new vehicles and removed parked ones
    Compute $T'(V^k)$ and $T''(V^k)$; ▷ Compute and update traveling times
    $x_k \leftarrow \text{PAP}(n^k, P, T', T'', C)$; ▷ Solve PAP($k$) with updated input
  end for
  return $f^{\text{DPAP}}(A)$ ▷ Compute the DPAP value

Initialization. The DPAP Algorithm 1 requires three input parameters: the set of parking lots P, where $|P| = m$, the residual capacity matrix $C = (c_{jt}), \forall j, t$ and the number of decision moments h. Note that, for simplicity, we note the set of parking lots with P, instead of $P(\Pi)$, for some policy Π (see Section 4.3). 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^1, which contains vehicle positions and destinations. According to their coordinates we compute the time needed to reach each parking $\big(T' = (t'_{ij}), i \in V^1, j \in P_i\big)$, and from each parking their destinations $\big(T'' = (t''_{ji}), i \in V^1, j \in P_i\big)$. We then solve the PAP(1), i.e., (1)–(4) where $k = 1$, for the given input, considering the associated

input denoted by n, P, T', T'', C.

Simulation loop. Once the solution of the static PAP(1) has been computed (line 5), we update the coordinates of the vehicles, based on the previous solution x, line 7. Vehicles that have arrived at their parking lots are stored in the list of parked vehicles 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^k (line 9). The travel time matrices, T' and T'', are then computed for the remaining vehicles V^k (line 11). Observe that vehicle speeds do not have to be homogeneous, i.e., different vehicle speeds will produce different driving times T' at each decision moment $k, 1 \leq k \leq h$. These driving times will then be supplied to the static PAP (1)–(4) (line 12) that will produce the parking-vehicle allocations for that decision moment k. The DPAP layer is responsible for providing the static PAP with the vehicle set and itself is not optimizing allocations. Therefore, before launching PAP($k+1$), the corresponding driving times T' will be computed taking into consideration the previous decision x_k.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 (5).

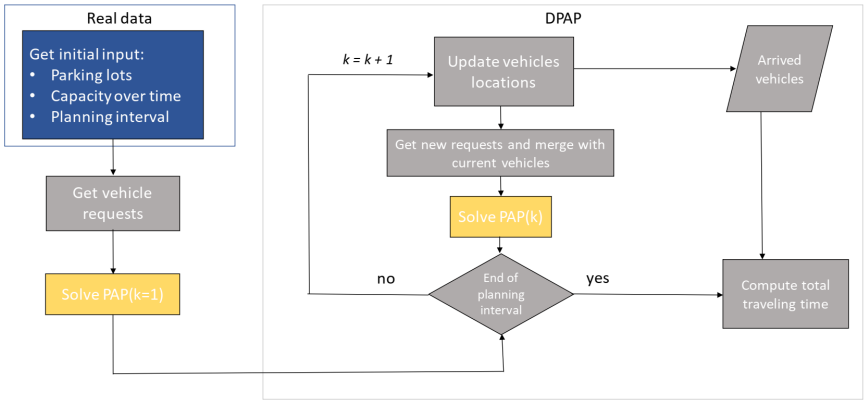

Figure 2: DPAP flowchart

Figure 2 depicts the flowchart of the DPAP procedure. 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.

3.3. Greedy heuristic

The final allocations of the DPAP does not guarantee optimality in the overall planning horizon. Therefore we wish to investigate how a heuristic would cope. Greedy heuristics represent a straightforward way to attain a feasible solution for most combinatorial problems. Our greedy heuristic is similar to the one proposed in [12] and is presented in Algorithm 2.

Algorithm 2 Greedy Add Algorithm

procedure GREEDY_ADD(k, n^k, m, C, T', T'')
  $U \leftarrow C$ $(u(j,t) \leftarrow c(j,t),\ \forall j,t)$; ▷ Copying capacity matrix $C = (c_{jt})$
  $T^* \leftarrow T' + (T'')^{\mathrm{T}}$ $(t^*(i,j) \leftarrow t'(i,j) + t''(j,i),\ \forall i,j)$; ▷ Computing total travel times
  $O \leftarrow$ Sort T^*; ▷ Sorting every row of T^*, storing it into order matrix $O_{n^k \times m} = o(i,j)$
  $f_{cur} \leftarrow 0$; ▷ Current value of the objective function f_{cur}
  for $i \leftarrow 1$ **to** n^k **do** ▷ For all active vehicles i
    for $j \leftarrow 1$ **to** $m+1$ **do** ▷ For all parking lots, including the dummy parking lot
      $l \leftarrow o(i,j); t \leftarrow t'(i,l) + t^k$; ▷ Next closest parking lot l of vehicle i; update arrival time t
      if $u(l,t) > 0$ **then** ▷ Check if there is an available spot at parking l in time t
        $x(i) \leftarrow l$; ▷ If true, assign i to parking l
        break; ▷ If the allocation was successful skip to the next vehicle $i+1$
      end if
    end for
    $u(l,t) \leftarrow u(l,t) - 1$; ▷ Reduce the capacity by 1 at time step t
    $f_{cur} \leftarrow f_{cur} + t'(i,l) + t''(l,i)$; ▷ Update the current objective function value f_{cur}
  end for
end procedure

We first copy input matrix C to get the current number of available spaces U at each parking and at each time unit. The array $x = (x_1, \ldots, x_{n^k})$ denotes the solution, where $x(i) = j$ means that vehicle i is assigned to parking j. In the case of the PAP, a constructive greedy heuristic can be based on sorting the parking lot by traveling times $(t^*_{ij} = t'_{ij} + t''_{ji},\ j = 1, \ldots, m)$ of all the vehicles i. This forms a matrix denoted by $O = (o_{ij}), i \in V^k, j \in P$ of vehicles sorted by their total traveling times. The Greedy heuristic then attempts to allocate a parking lot to a vehicle i in the order starting from its closest parking (o_{i1}), second closest (o_{i2}), etc. We present an example to illustrate the greedy solutions and get an insight of its advantages and weaknesses.

Example. Consider a set of $m = 3$ parking lots and a set of $n = 5$ vehicles at the decision moment

t_0, over a time horizon of five time steps, i.e. $h = 5$. Suppose the driving and walking times are

$$T' = \begin{bmatrix} 5 & 1 & 5 & 0 \\ 1 & 5 & 4 & 0 \\ 3 & 3 & 3 & 2 \\ 3 & 2 & 3 & 1 \\ 1 & 4 & 3 & 2 \end{bmatrix} \text{ and } (T'')^{\mathrm{T}} = \begin{bmatrix} 8 & 3 & 5 & 100 \\ 3 & 7 & 4 & 100 \\ 6 & 1 & 5 & 100 \\ 6 & 3 & 6 & 100 \\ 4 & 5 & 2 & 100 \end{bmatrix},$$

respectively. Note that, the fourth column in matrices T' and $(T'')^{\mathrm{T}}$ represents the dummy facility, where the walking times T'' are penalized by $M = 100$ time steps ($M >> h$).

In our example, we distinguish two types of capacities (see Section 5) denoted by $C^{(reg)}$ and $C^{(red)}$. The first one corresponds to what we call *regular capacity* in which the demand does not surpass the supply. The second, $C^{(red)}$ corresponds to the case where we do not know if there is a sufficient supply. We call this case *reduced capacity*.

$$C^{(reg)} = \begin{bmatrix} 1 & 2 & 1 & 1 & 3 \\ 1 & 1 & 2 & 0 & 3 \\ 2 & 2 & 2 & 2 & 2 \\ 10 & 10 & 10 & 10 & 10 \end{bmatrix}, \quad C^{(red)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 10 & 10 & 10 & 10 & 10 \end{bmatrix}.$$

Note that, the last row of $C^{(reg)}$ and $C^{(red)}$ represents the dummy facility capacity. We can observe that the optimal solutions will be

$$x^*_{(reg)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \text{ and } x^*_{(red)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

with total traveling times of 22 and 216, respectively (see objective function (1)). In the case of reduced capacity, vehicles 3 and 4 will not find an available slot, and will be directed to their destinations.

We will now provide the solution of the greedy heuristic at the decision moment t_0. The corresponding order matrix O is obtained by sorting each row of the total traveling time matrix

T^* by column (parking) in non-decreasing order, where values in parentheses correspond to the traveling time t^*_{ij}:

$$O = \begin{bmatrix} 2^{(4)} & 3^{(10)} & 1^{(13)} & 4^{(100)} \\ 1^{(4)} & 3^{(8)} & 2^{(12)} & 4^{(100)} \\ 2^{(4)} & 3^{(8)} & 1^{(9)} & 4^{(102)} \\ 2^{(5)} & 1^{(9)} & 3^{(9)} & 4^{(101)} \\ 1^{(5)} & 3^{(5)} & 2^{(9)} & 4^{(102)} \end{bmatrix}.$$

The greedy heuristic will then attempt to assign a parking lot to each vehicle in that order. For $C^{(reg)}$, the first attempt to allocate a vehicle to it's closest parking (column one of matrix O) will be successful, thus $x^{\text{greedy}}_{(reg)} = x^*_{(reg)}$. For $C^{(red)}$, the greedy algorithm will assign the first three vehicles to their closest parking lot, parking 2, 1 and 3 respectively. The last two vehicles will be directed to their destinations at t_0. This solution yields an objective of 219, which is not optimal.

At the following decision moment t_1, the driving times T' will be accordingly updated and the vehicle that arrived will be removed (added to the list A) and n^1_{new} new requests will be added.

Now assume that a driver follows the GPS guiding and behaves as follows: (i) s/he drives to the parking lot j' closest to his destination, consuming $t'_{ij'}$ time; (ii) if it is occupied, the next closest lot j'' is attempted, consuming an additional time $t_{j'j''}$; (iii) if parking j'' is occupied, s/he continues to the third closest, and so on. We call this search strategy (algorithm) the usual driver strategy (UDS).

Property 1. *The solution obtained by the UDS is not better than the greedy heuristic.*

Proof. Without loss of generality, assume that vehicle i reaches its parking lot after one failure, i.e., assume that $j = j''$. The time Greedy spends to reach it is t'_{ij}. The time spent by the UDS algorithm is $t'_{ij'} + t'_{j'j}$. The result then holds due to the triangular inequality and because $t'_{j'j} > 0$. □

We can conclude that the Greedy algorithm will provide a better solution than the UDS, which can represent an estimation of real drivers choices. This is why in the following sections we will consider both the Greedy and Exact solutions in steps 5 and 12 of the Algorithm 1 to solve the static PAP.

4. 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.

4.1. Collected data

We found three cities with more than 500,000 inhabitants and with accessible real-time parking availability information online: Belgrade (Serbia)[2], Luxembourg (Luxembourg)[3] and Lyon (France)[4].

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.

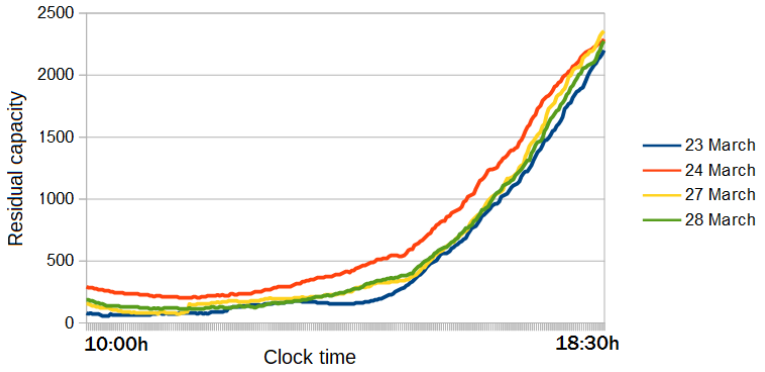

Figure 3: Availability change over time for March 2017 for the Bouillon parking in Luxembourg.

With regard to the central theme of this study, parking allocation in peak hours, we set apart

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

the weekdays and the weekends. We note (see Figure 3) that a similar behavior can be observed during weekdays when most of the traffic congestion occurs, while the weekends show less vehicle activity. Of course, 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.).

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.

4.2. Simulation parameters

We first introduce the random variables needed to simulate the dynamic parking process for the time interval H. 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-word 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.

4.2.1. 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 4 for the map of Lyon). This area is represented by 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 S of this area, is computed as the arithmetical mean of all the parking facilities coordinates, more precisely

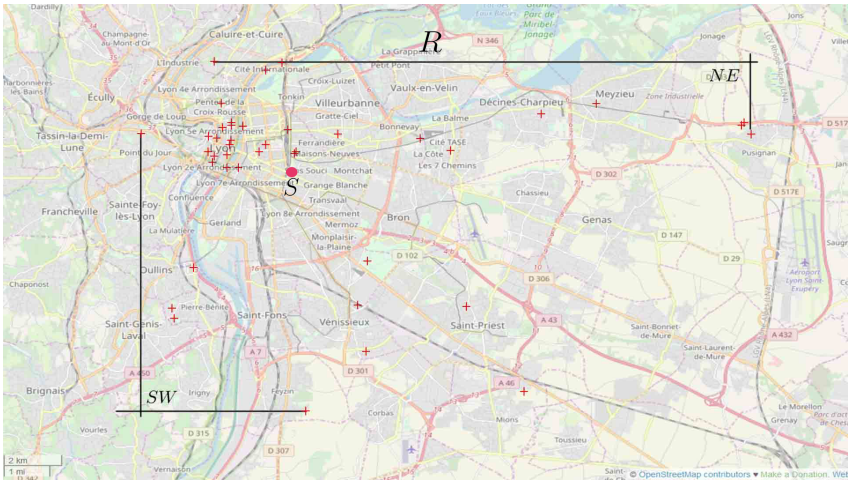

Figure 4: Map of the city of Lyon. The red crosses represent parking lots.

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

This approximation has proven to be an accurate estimation of the real city center, since the majority of parking facilities are centered in dense urban areas (in the case of the cities we examined).

4.2.2. New vehicles

To achieve 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,

$$n^k_{new} = \begin{cases} 0, & \text{if } \sum\limits_{j \in P \setminus \{m+1\}} \big[c_{jt_k} - c_{jt_{k-1}}\big] \leq 0 \\ \Big\lceil \gamma \sum\limits_{j \in P \setminus \{m+1\}} \big[c_{jt_k} - c_{jt_{k-1}}\big] \Big\rceil, & \text{else} \end{cases}$$

for some $\gamma \in \mathbb{R}^+$.

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. The vertical axis represents the number of requests, the horizontal axis represents the clock time, from 00:00 to 23:59, divided into 1,440 one-minute time steps.

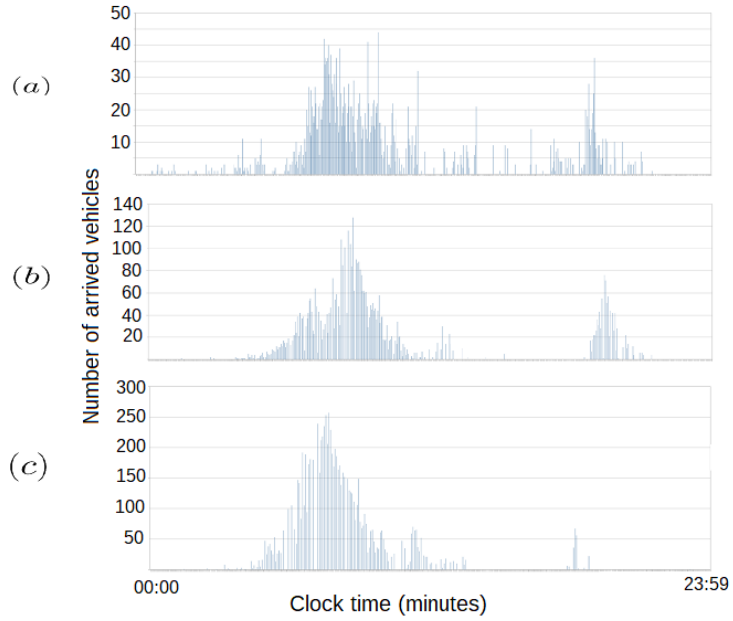

Figure 5: Request inflow diagram for the cities of: (a) Belgrade, (b) Luxembourg and Lyon.

Firstly, we set the vehicles speed to be constant at 30 km/h and the walking speed to be 6 km/h. Modifying these values does not increase the complexity of the DPAP framework. Section 5 provides more results and observations when modifying some of these parameters.

4.2.3. 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.

$$(lat_i^-, lon_i^-) \sim \mathcal{U}(R) \quad \forall i \in V^k_{\text{new}},$$

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

Their destinations will be centered near the city center S. That is why we set the destinations to be determined by a normal distribution centered around S. The standard deviation σ^2, if not specified otherwise, is 0.15.

$$(lat_i^+, lon_i^+) \sim \mathcal{N}(S, \sigma^2) \quad \forall i \in V_{\text{new}}^k. \tag{6}$$

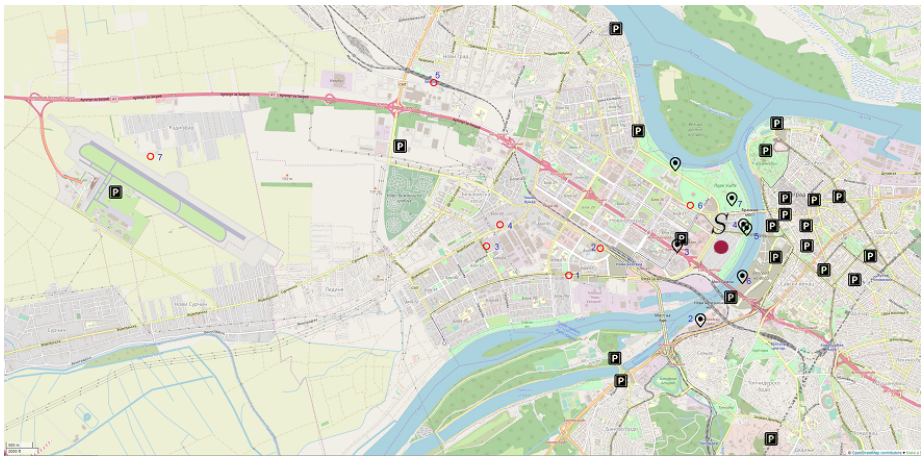

Figure 6: An example of vehicle generation on the Belgrade map.

An example of generating seven vehicles on the map of the city of Belgrade is presented in Figure 6, where the red circles represent vehicles positions and the vehicles numbers, the pinpoints represent vehicle destinations. 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.

4.3. Policies

The policies are the strategic guidelines at the top layer of our framework introduced in Section 2. The main 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 paper, 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 and vehicle i.

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 Δ_k this policy forms the 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 depend on the decision moment Δ_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 Δ_k, will not be greater at each subsequent PAP$(l), l > k$, hence enlarging the set $P_i(\Pi_2)$ over time.

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. Each policy Π_i depends on a parameter α_i, $i = 1, 2, 3$. To simplify the notation we refer to a specific parameter and a policy as $\alpha(\Pi_i)$.

5. 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. In the next subsections we report the mean results observed over 1,050 different instances generated to evaluate our framework under different

configurations. We start with experiments to evaluate the behavior of our system and to compare the performance of the greedy algorithm and that of an exact solution procedure when considering different scenarios.

5.1. No-restriction policy

In this subsection we investigate the DPAP where vehicles can be allocated to any parking lot, i.e., no policies are applied. These tests will allow us to gain an insight into the performance of the DPAP, because its complexity mostly relies on the static model. For example, we would like to know what will be the effect at the end of the planning horizon if the greedy algorithm is applied compared to an exact solution. As our experiments will demonstrate, if there are sufficient parking slots to accommodate all the vehicles, the greedy algorithm provides near-optimal solutions. The more compelling scenario is when capacities are reduced, or equivalently the number of vehicles is increased. This is why we have divided our instances into two groups: with standard and reduced residual capacities c_{jt}. The results are also divided by cities. For each city we report their total number of parking lots, the number of arrived vehicles $|A|$ during the planning horizon, the CPU execution times and the number of changed allocations, in their respective columns. The column *Number of changes* counts the total number of parking reallocations over the planning horizon. Note that, in Table 2 the objective f^{DPAP} is presented, while Table 3 compares the number of unparked vehicles instead. This is because the we can use f^{DPAP} only if all vehicles are assigned a parking lot. Tables 3 and 4 last column represents the difference between the number of unparked vehicles of the model using the exact and greedy algorithms.

			Execution time		**Number of changes**		$f^{\mathbf{DPAP}}$		
	m	$\|A\|$	Exact	Greedy	Exact	Greedy	Exact	Greedy	Difference
Belgrade	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 2: Instances with standard capacities where $P_i = P$ for all vehicles i.

From Table 2 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 execution times for solving the model 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	$\|A\|$	Exact	Greedy	Exact	Greedy	Exact	Greedy	Difference
Belgrade	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 3: Instances with reduced capacities where $P_i = P$ for all vehicles i.

However, Table 3 reveals that when the parking availability becomes limited, the exact algorithm presents clear advantages. More precisely, over time the vehicle sets $V^k(greedy)$ and $V^k(exact)$ will diverge and produce different driving times t'_{ij}. 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. We observe that this case has a particular impact on smaller maps, as was the case for Luxembourg, where the average difference was 81.7. Furthermore, we encounter a significantly higher number of changes when compared with the standard capacity, see column *Number of changes* in Table 3. Namely, exact algorithm 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 largest map 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 over the planning horizon H. The average Exact 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 the model, and 2.5% more with the greedy algorithm. Similar results can be observed for Belgrade and Luxembourg, see Table 4.

From Tables 2, 3 and 4 we can conclude that Exact algorithm constitutes the better option, both in terms of the solution quality and execution time. However, we cannot guarantee that some

			Execution time		Unparked		
	m	$\lvert A \rvert$	Exact	Greedy	Exact	Greedy	Difference
Belgrade	23	64,680	499.0	1.6	29078	31361	7.3%
Luxembourg	18	134,660	307.0	1.0	109232	111187	1.8%
Lyon	47	213,660	5840.4	17.1	149396	154757	2.5%

Table 4: Instances with reduced capacities and vehicle overload, $\gamma = 20$.

vehicles will not be allocated to an unfavorable parking lot, just because there is an available slot at their arrival time. To remedy this situation, we introduce policies to the DPAP mechanism and analyze their impact on the solution.

5.2. Policy analysis

The aim of this subsection is to evaluate the effect policies display over the DPAP framework. In order to illustrate the impact of the parameter on the policy, 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 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 7. More precisely, we selected values of α in the range where the slope of the curves in Figure 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 destination. For the maximal traveling time policy, we set these values between 20 and 50 minutes. 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 5.1, two scenarios were considered: with reduced and with regular parking capacities.

Table 5 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 total number

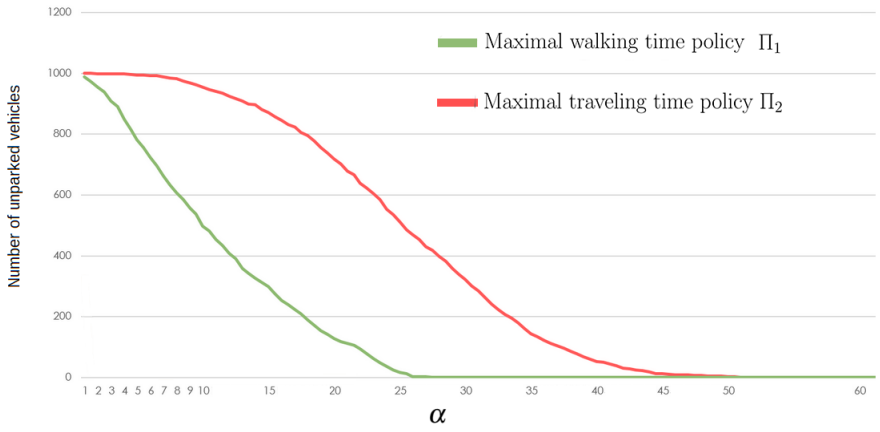

Figure 7: The impact of the policy parameter α on the number of unparked vehicles.

of changes represents the number of times vehicles have been reallocated from one parking lot to another, depending on the current traffic structure. These values are presented horizontally in rows *Reallocations* in Table 5. The table is divided horizontally by cities and vertically by policies. The results are then reported for each value of the parameter α. The rows *difference* in Table 5 represent the percentage difference between the regular and reduced capacity cases, 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-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 reallocations 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

			Belgrade				Luxembourg				Lyon			
			$m = 23$			$\|A\| = 3234$	$m = 18$			$\|A\| = 6733$	$m = 47$			$\|A\| = 10683$
			$\alpha = 10$	$\alpha = 20$	$\alpha = 25$	$\alpha = 30$	$\alpha = 10$	$\alpha = 20$	$\alpha = 25$	$\alpha = 30$	$\alpha = 10$	$\alpha = 20$	$\alpha = 25$	$\alpha = 30$
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
		Reallocations	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
		Reallocations	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
		Reallocations	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
		Reallocations	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
		Reallocations	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
		Reallocations	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
		Reallocations	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
		Reallocations	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%
		Reallocations	178.1%	171.9%	175%	180.1%	196.9%	197%	196.6%	197.2%	145.2%	156.4%	162.6%	173.2%

Table 5: Policy comparison for Belgrade, Luxembourg and Lyon, for four preset values of the parameter α.

corresponding to Figure 7.

Comparison per policy. From Table 5 we recognize that Π_3 is the least able to cope with reduced capacities. This can be observed by the biggest difference in the number of unparked vehicles when comparing regular and reduced capacities. 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 *Reallocations* in Table 5) 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 Lyon. 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 91%. From Table 5 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. This first set of experiments demonstrates the capability of our framework to manage a large number of vehicles in a dynamic manner. In the next subsection we present other experiments to evaluate the impact of different parameters on the framework.

5.3. Additional experiments

In the following, we present additional results to demonstrate the effectiveness of our framework in particular configurations. More precisely, we first discuss several points related to the drivers destinations, such as the impact of their distribution on our DPAP framework. We then focus on the impact of traffic by introducing variable speeds during our experiments.

5.3.1. Discussion about destinations

The drivers destinations are used by our DPAP framework to determine the best parking lot for each driver according to the underlying walking time. We computed the percentage of vehicles that will be allocated to the parking lot closest to their destination. These data provide us additional insights about the overall quality of individual allocations and assure the users that the DPAP should provide satisfactory and fair outcomes, even without introducing policies Π. We observe that on average 58.4% of vehicles are indeed allocated to the parking lot closest to their destinations considering regular capacities. In the case of reduced capacities this value drops to 48.2%. Moreover, the maximal walking distance will be 775 meters and 1,563 meters on the biggest map of Lyon for regular and reduced capacities, respectively. All the numerical results represent the mean values over ten runs.

When no parking can be offered at the current decision moment the vehicles are guided towards their destinations. This can lead to time lost cruising. Actually, the DPAP framework proved to keep the vehicles cruising around four to five minutes on average under heavy traffic overload and reduced capacities as presented in Table 6. This further demonstrates that the DPAP does not only provide quick solutions per decision moment, but that it also brings a clear benefit to the drivers.

In the results presented in the previous sections, the drivers destinations were centered around a narrow area around the city center. Recall that the destinations, obtained by (6), are determined

	Belgrade	**Luxembourg**	**Lyon**
Vehicle number	32340	13466	32049
Number of unparked	5110	1956	3491
Average time spent in dummy	4.0 minutes	5.0 minutes	4.0 minutes

Table 6: Average time spent cruising.

by a normal distribution centered around the city center S, i.e., $\sigma^2 = 0.15$. Increasing this value corresponds adds more variability to their destinations. Table 7 shows the evolution of the average number of reallocation obtained by the DPAP framework, as well as the average walking and driving times when the standard deviation changes. These results are calculated with an average of 10 simulations over the city of Belgrade.

	Regular capacity			**Reduced capacity**		
σ^2	Avg reallocations	Avg driving time	Avg walking time	Avg reallocations	Avg driving time	Avg walking time
0.05	0.2	12.5	7.3	1704.8	12.8	8.0
0.1	7.2	12.9	7.0	1084.2	13.1	7.5
0.15	14.4	13.1	8.1	874.0	13.2	8.3
0.2	20.1	13.4	9.7	839.1	13.5	9.9
0.25	28.3	13.6	11.9	799.1	13.6	12.0
0.30	42.0	13.7	14.9	709.7	13.8	14.8
0.35	47.2	13.9	18.2	912.3	13.9	18.3
0.4	48.6	14.1	22.0	891.8	14.2	21.9
0.45	59.2	14.2	26.0	990.1	14.4	25.9
0.5	67.1	14.3	30.5	1007.9	14.4	30.2
Average	33.43	13.57	15.56	981.3	13.69	15.68

Table 7: Results with varying destinations deviation parameter σ^2 for the city of Belgrade with 3,234 vehicles.

When we modify the deviation parameter σ^2, we first observe an expected outcome. The average driving and walking time will rise, as σ^2 becomes greater. This is because the vehicle's destination can vary more as σ^2 increases. Moreover, the majority of parking facilities are near the city center as shown in Figures 4 and 6. Hence, the walking times are significantly higher the further the destination is from the center. Of course, when the destination is really far away from the city center, the users are less likely to rely on the DPAP, specially when most parking lots are located in the city center, as is the case for the city of Belgrade (see Figure 6). However, we observe that the

driving and walking times are not really impacted when capacities are reduced, when compared to regular capacities. This is achieved by enabling parking reallocations during the planning horizon, as shown in Tables 2–4 and 7. These results demonstrate the DPAP mechanism is able to cope with reduced capacities with little to none detriment to the users, regardless of their destinations.

5.3.2. Impact of traffic

In the previous sections, we have intensively evaluated the scalability of our framework. We particularly considered configurations with a low number of available places and a high number of requests. These conditions indeed correspond to situations where a system like ours can prove to be the most useful. However, when the number of vehicles searching for a parking space is high, the impact on traffic is not negligible. Driving times to parking lots can then significantly increase. This phenomenon can obviously be limited if the allocation system is working properly but it may arise and so we assess its impact on our DPAP framework. We introduce variable speeds that reflect the congestion effect. In order to approximate the speed of the vehicles at the decision moment Δ_k and thus their driving times t'_{ij} (denoted $t'_{ij}(k)$), we use the formula provided by [31]:

$$t'_{ij}(k) = \text{Link travel time}_{ij}(k) = free\_flow\_time_{ij} * \left(1 + \frac{flow_j(k)}{q_j}\right)^{Pow}, \; i \in V^k, j \in P, k \in H. \tag{7}$$

The $free\_flow\_time_{ij}$ is set to be the time that the vehicle i would take to reach its assigned parking lot j at the speed of 30 km/h. The $flow_j(k)$ is the number of all the vehicles assigned to the parking j at the time step $k-1$ with a total capacity of q_j, i.e., $flow_j(k) = \sum_{i \in V^{k-1}} x_{ij}(k-1)$. The parameter Pow serves to further increase or decrease the effect of the vehicle inflow. Based on [31] and their repository[5], we set the value of Pow to be four. By computing the driving times this way, the PAP model remains linear and its complexity does not change.

	Constant speed		Variable speed		
Capacity	Avg driving	Avg walking	Avg driving	Avg walking	Different lots (%)
Regular	15.1	11.0	24.3	18.7	3422.4 (32%)
Reduced	16.0	14.9	24.3	16.3	4,365 (41%)

Table 8: Results with constant and variable speeds on the city of Lyon with 10,683 vehicles.

[5]https://github.com/bstabler/TransportationNetworks

In Table 8, we present the average driving and walking times for constant and variable speeds with reduced and regular capacities. These results are obtained with the default value of $\sigma^2 = 0.15$. The *Different lots* column reports the average number of vehicles that were assigned to different lots when traveling at constant or varying speed. When the vehicles speed is reduced due the the congestion effect (7), the driving time increases by around eight minutes in average. However, the average walking time improves by 2.4 minutes when capacities are reduced. As shown previously, this is made possible by reallocating vehicles as data are updated.

We also notice that vehicles will mostly be parked at the same parking lots as they would have been if traveling at constant speed: 68% when capacity is regular and 59% when reduced. The overall utilization of each parking lot remains very similar in both cases, as shown in Figures 8 and 9. When the effects of congestion and its consequences on vehicle speed is included, we observe a more balanced distribution of parking use. The best example is parking 14 in Figure 8 that would clearly have been overcrowded if the congestion effects were not taken into consideration. However, when parking capacities are low, we observe that taking traffic congestion into consideration would not significantly change the parking utilization rate as shown in Figure 9, since there are already very limited options. We can conclude that the DPAP proves to be advantageous in cases where we would want to balance the parking utilization considering potential congestion it could cause.

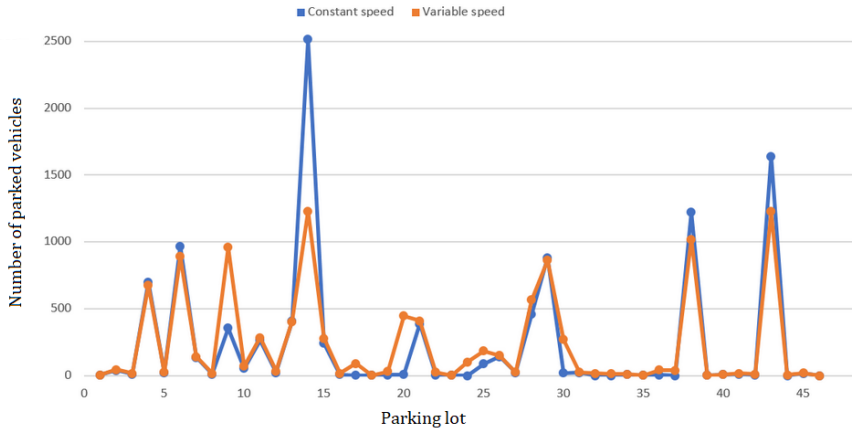

Figure 8: Utilization of parking lots in Lyon for constant and variable speeds with regular capacity.

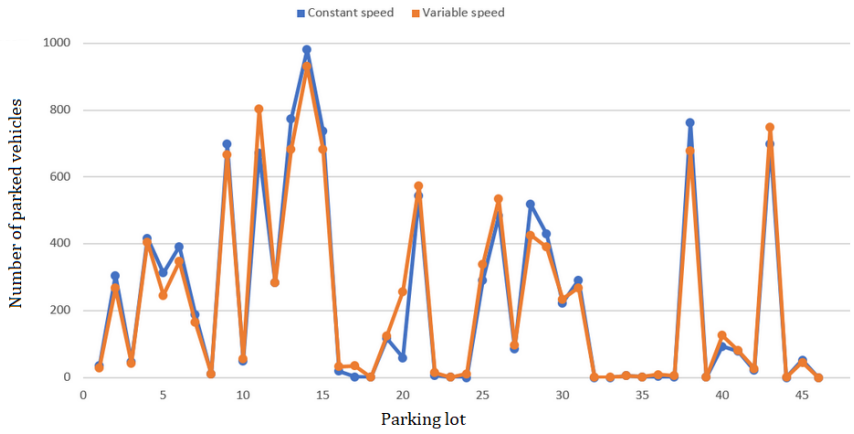

Figure 9: Utilization of parking lots in Lyon for constant and variable speeds with reduced capacity.

6. Conclusions & future work

We have proposed a framework to solve the parking allocation problem (PAP) for connected vehicles over a given time period. We have adapted a recently published static parking allocation model for connected vehicles, proposed in [12], since its flexibility allows it to cope with the dynamic changes that can frequently occur over the planning horizon.

This static model was solved at each time step 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 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 the 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. Moreover, the DPAP framework scales easily with a large number of vehicles and provides a robust solution for various scenarios, maintaining similar traveling times for users even when the capacities are reduced.

We have demonstrated that the dynamic PAP framework could provide a good basis for a parking allocation system and could be further refined to provide even more accurate and fair parking allocations to users. Introducing a more explicit approach can be implemented where the exogenous vehicle set and compliance is less likely. A game-theoretic approach can be added to our framework to maintain an equilibrium over the planning horizon. To improve the static optimization results we could include a double horizon heuristic that would keep track of the effect of the current decision and adapt future ones, e.g., [32]. This would provide more accurate allocations, especially when the vehicle speed is not considered to be constant.

Acknowledgments

This work was partly funded by the Canadian National Sciences and Engineering Research Council under grant 2015-06189 and the Hauts-de-France region. Their support is gratefully acknowledged. We are grateful to the referees for their valuable comments and suggestions which helped us to improve the paper.

References

[1] D. C. Shoup, High cost of free parking, Journal of Planning Education and Research 17 (1997) 3–22.

[2] D. C. Shoup, Cruising for parking, Transport Policy 13 (6) (2006) 479–486.

[3] E. Gantelet, A. Lefauconnier, The time looking for a parking space: Strategies, associated nuisances and stakes of parking management in France, Europe Transport Conference, 2006, pp. 1–6.

[4] T. Lin, H. Rivano, F. Le Mouel, A survey of smart parking solutions, IEEE Transactions on Intelligent Transportation Systems 18 (12) (2017) 3229–3253.

[5] D. Shoup, The High Cost of Free Parking, Chicago: Planners Press, 2011.

[6] F. Caicedo, H. Lopez-Ospina, R. Pablo-Malagrida, Environmental repercussions of parking demand management strategies using a constrained logit model, Transportation Research Part D: Transport and Environment 48 (2016) 125–140.

[7] M. Florian, M. Los, Impact of the supply of parking spaces on parking lot choice, Transportation Research Part B: Methodological 14 (1–2) (1980) 155–163.

[8] V. Bayram, B. Tansel, H. Yaman, Compromising system and user interests in shelter location and evacuation planning, Transportation Research Part B: Methodological 72 (2015) 146–163.

[9] E. Angelelli, I. Arsik, V. Morandi, M. Savelsbergh, M. Speranza, Proactive route guidance to avoid congestion, Transportation Research Part B: Methodological 94 (2016) 1–21.

[10] E. Angelelli, V. Morandi, M. Speranza, A trade-off between average and maximum arc congestion minimization in traffic assignment with user constraints, Computers & Operations Research 110 (2019) 88–100.

[11] S. Abidi, S. Krichen, E. Alba, J. M. M. Bravo, A hybrid heuristic for solving a parking slot assignment problem for groups of drivers, International Journal of Intelligent Transportation Systems Research 15 (2) (2017) 85–97.

[12] M. Mladenović, T. Delot, G. Laporte, C. Wilbaut, The parking allocation problem for connected vehicles, Journal of Heuristics 26 (2020) 377–399.

[13] C. Shao, H. Yang, Y. Zhang, J. Ke, A simple reservation and allocation model of shared parking lots, Transportation Research Part C: Emerging Technologies 71 (2016) 303–312.

[14] C. Tang, X. Wei, C. Zhu, W. Chen, J. J. P. C. Rodrigues, Towards smart parking based on fog computing, IEEE Access 6 (2018) 70172–70185.

[15] M. Roca-Riu, E. Fernández, M. Estrada, Parking slot assignment for urban distribution: Models and formulations, Omega 57, Part B (2015) 157–175.

[16] V. Verroios, V. Efstathiou, A. Delis, Reaching available public parking spaces in urban environments using ad hoc networking, in: Mobile Data Management (MDM), 2011 12th IEEE International Conference on, Vol. 1, IEEE, 2011, pp. 141–151.

[17] M. Mladenović, The dynamic parking allocation problem: theoretical and practical solution methods, Ph.D. thesis, Université Polytechnique Hauts-de-France (2020).

[18] J. Cavadas, G. Homem de Almeida Correia, J. Gouveia, 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 (2015) 188–201.

[19] A. Millard-Ball, The autonomous vehicle parking problem, Transport Policy 75 (2019) 99–108.

[20] H. N. Psaraftis, M. Wen, C. A. Kontovas, Dynamic vehicle routing problems: Three decades and counting, Networks 67 (1) (2016) 3–31.

[21] V. Pillac, M. Gendreau, C. Guéret, A. L. Medaglia, A review of dynamic vehicle routing problems, European Journal of Operational Research 225 (1) (2013) 1–11.

[22] G. Berbeglia, J.-F. Cordeau, G. Laporte, Dynamic pickup and delivery problems, European Journal of Operational Research 202 (1) (2010) 8–15.

[23] M. S. Farag, M. M. Mohie El Din, H. A. El Shenbary, Smart parking guidance using optimal cost function, Computer and Information Science 10 (1) (2017) 48–53.

[24] K. B. Dsouza, S. Mohammed, Y. Hussain, Smart parking—An integrated solution for an urban setting, in: Convergence in Technology (I2CT), 2017 2nd International Conference, IEEE, 2017, pp. 174–177.

[25] T. Delot, S. Ilarri, S. Lecomte, N. Cenerario, Sharing with caution: Managing parking spaces in vehicular networks, Mobile Information Systems 9 (1) (2013) 69–98.

[26] T. Delot, N. Cenerario, S. Ilarri, S. Lecomte, A cooperative reservation protocol for parking spaces in vehicular ad hoc networks, in: Proceedings of the 6th International Conference on Mobile Technology, Application & Systems, ACM, 2009, pp. 1–8.

[27] Y. Geng, C. Cassandras, Dynamic resource allocation in urban settings: A smart parking approach, in: 2011 IEEE International Symposium on Computer-Aided Control System Design (CACSD), 2011, pp. 1–6.

[28] J. Toutouh, E. Alba, Distributed fair rate congestion control for vehicular networks, in: Distributed Computing and Artificial Intelligence, 13th International Conference, no. 474 in Advances in Intelligent Systems and Computing, Springer, Cham, 2016, pp. 433–442.

[29] B. Zou, N. Kafle, O. Wolfson, J. J. Lin, A mechanism design based approach to solving parking slot assignment in the information era, Transportation Research Part B: Methodological 81 (2015) 631–653.

[30] R. G. Thompson, P. Bonsall, Drivers' response to parking guidance and information systems, Transport Reviews 17 (2) (1997) 89–104.

[31] O. Jahn, R. H. Möhring, A. S. Schulz, N. E. Stier-Moses, System-optimal routing of traffic flows with user constraints in networks with congestion, Operations research 53 (4) (2005) 600–616.

[32] S. Mitrović-Minić, R. Krishnamurti, G. Laporte, Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows, Transportation Research Part B: Methodological 38 (8) (2004) 669–685.

***Credit Author Statement**

Term	Definition	Authors
Conceptualization	Ideas; formulation or evolution of overarching research goals and aims	M. Mladenovic, T. Delot, G. Laporte, C. Wilbaut
Methodology	Development or design of methodology; creation of models	M. Mladenovic, T. Delot, G. Laporte, C. Wilbaut
Software	Programming, software development; designing computer programs; implementation of the computer code and supporting algorithms; testing of existing code components	M. Mladenovic
Validation	Verification, whether as a part of the activity or separate, of the overall replication/ reproducibility of results/experiments and other research outputs	M. Mladenovic
Formal analysis	Application of statistical, mathematical, computational, or other formal techniques to analyze or synthesize study data	Mladenovic, Delot, Laporte, Wilbaut
Investigation	Conducting a research and investigation process, specifically performing the experiments, or data/evidence collection	M. Mladenovic
Resources	Provision of study materials, reagents, materials, patients, laboratory samples, animals, instrumentation, computing resources, or other analysis tools	M. Mladenovic
Data Curation	Management activities to annotate (produce metadata), scrub data and maintain research data (including software code, where it is necessary for interpreting the data itself) for initial use and later reuse	M. Mladenovic
Writing - Original Draft	Preparation, creation and/or presentation of the published work, specifically writing the initial draft (including substantive translation)	M. Mladenovic, T. Delot, G. Laporte, C. Wilbaut
Writing - Review & Editing	Preparation, creation and/or presentation of the published work by those from the original research group, specifically critical review, commentary or revision – including pre-or postpublication stages	M. Mladenovic, T. Delot, G. Laporte, C. Wilbaut
Visualization	Preparation, creation and/or presentation of the published work, specifically visualization/ data presentation	M. Mladenovic
Supervision	Oversight and leadership responsibility for the research activity planning and execution, including mentorship external to the core team	T. Delot, G. Laporte, C. Wilbaut
Project administration	Management and coordination responsibility for the research activity planning and execution	T. Delot, G. Laporte, C. Wilbaut
Funding acquisition	Acquisition of the financial support for the project leading to this publication	T. Delot, G. Laporte, C. Wilbaut