Springer

Dear Author,

Here are the proofs of your article.

- You can submit your corrections **online**, via **e-mail** or by **fax**.
- For **online** submission please insert your corrections in the online correction form. Always indicate the line number to which the correction refers.
- You can also insert your corrections in the proof PDF and **email** the annotated PDF.
- For fax submission, please ensure that your corrections are clearly legible. Use a fine black pen and write the correction in the margin, not too close to the edge of the page.
- Remember to note the **journal title**, **article number**, and **your name** when sending your response via e-mail or fax.
- **Check** the metadata sheet to make sure that the header information, especially author names and the corresponding affiliations are correctly shown.
- **Check** the questions that may have arisen during copy editing and insert your answers/ corrections.
- **Check** that the text is complete and that all figures, tables and their legends are included. Also check the accuracy of special characters, equations, and electronic supplementary material if applicable. If necessary refer to the *Edited manuscript*.
- The publication of inaccurate data such as dosages and units can have serious consequences. Please take particular care that all such details are correct.
- Please **do not** make changes that involve only matters of style. We have generally introduced forms that follow the journal's style.
  Substantial changes in content, e.g., new results, corrected values, title and authorship are not allowed without the approval of the responsible editor. In such a case, please contact the Editorial Office and return his/her consent together with the proof.
- If we do not receive your corrections **within 48 hours**, we will send you a reminder.
- Your article will be published **Online First** approximately one week after receipt of your corrected proofs. This is the **official first publication** citable with the DOI. **Further changes are, therefore, not possible.**
- The **printed version** will follow in a forthcoming issue.

Please note

After online publication, subscribers (personal/institutional) to this journal will have access to the complete article via the DOI using the URL: http://dx.doi.org/[DOI].
If you would like to know when your article has been published online, take advantage of our free alert service. For registration and further information go to: http://www.link.springer.com.

Due to the electronic nature of the procedure, the manuscript and the original figures will only be returned to you on special request. When you return your corrections, please inform us if you would like to have these documents returned.

Metadata of the article that will be visualized in OnlineFirst

ArticleTitle	The parking allocation problem for connected vehicles	
Article Sub-Title		
Article CopyRight	Springer Science+Business Media, LLC, part of Springer Nature (This will be the copyright line in the final PDF)	
Journal Name	Journal of Heuristics	
Corresponding Author	Family Name	**Mladenović**
	Particle	
	Given Name	**Marko**
	Suffix	
	Division	
	Organization	UVHC, LAMIH UMR CNRS 8201
	Address	Mont Houy, 59313, Valenciennes, France
	Phone	+33 (0)3 27 51 12 34
	Fax	
	Email	marko.mladenovic@univ-valenciennes.fr
	URL	
	ORCID	http://orcid.org/0000-0002-8452-851X
Author	Family Name	**Delot**
	Particle	
	Given Name	**Thierry**
	Suffix	
	Division	
	Organization	UVHC, LAMIH UMR CNRS 8201
	Address	Mont Houy, 59313, Valenciennes, France
	Phone	
	Fax	
	Email	
	URL	
	ORCID	
Author	Family Name	**Laporte**
	Particle	
	Given Name	**Gilbert**
	Suffix	
	Division	
	Organization	HEC Montréal
	Address	3000, chemin de la Côte-Sainte-Catherine, Montreal, H3T 2A7, Canada
	Phone	
	Fax	
	Email	
	URL	

	ORCID	
Author	Family Name	**Wilbaut**
	Particle	
	Given Name	**Christophe**
	Suffix	
	Division	
	Organization	UVHC, LAMIH UMR CNRS 8201
	Address	Mont Houy, 59313, Valenciennes, France
	Phone	
	Fax	
	Email	
	URL	
	ORCID	

	Received	4 September 2017
Schedule	Revised	24 November 2017
	Accepted	19 December 2017

Abstract

In this paper, we propose a parking allocation model that takes into account the basic constraints and objectives of a problem where parking lots are assigned to vehicles. We assume vehicles are connected and can exchange information with a central intelligence. Vehicle arrival times can be provided by a GPS device, and the estimated number of available parking slots, at each future time moment and for each parking lot is used as an input. Our initial model is static and may be viewed as a variant of the generalized assignment problem. However, the model can be rerun, and the algorithm can handle dynamic changes by frequently solving the static model, each time producing an updated solution. In practice this approach is feasible only if reliable quality solutions of the static model are obtained within a few seconds since the GPS can continuously provide new input regarding the vehicle's positioning and its destinations. We propose a 0–1 programming model to compute exact solutions, together with a variable neighborhood search-based heuristic to obtain approximate solutions for larger instances. Computational results on randomly generated instances are provided to evaluate the performance of the proposed approaches.

Keywords (separated by '-'): Parking allocation - 0–1 Programming - Variable neighborhood search

Footnote Information

J Heuristics
https://doi.org/10.1007/s10732-017-9364-7

The parking allocation problem for connected vehicles

Marko Mladenović[1] · Thierry Delot[1] · Gilbert Laporte[2] · Christophe Wilbaut[1]

Received: 4 September 2017 / Revised: 24 November 2017 / Accepted: 19 December 2017

Abstract In this paper, we propose a parking allocation model that takes into account the basic constraints and objectives of a problem where parking lots are assigned to vehicles. We assume vehicles are connected and can exchange information with a central intelligence. Vehicle arrival times can be provided by a GPS device, and the [1] estimated number of available parking slots, at each future time moment and for each parking lot is used as an input. Our initial model is static and may be viewed as a [2] variant of the generalized assignment problem. However, the model can be rerun, and the algorithm can handle dynamic changes by frequently solving the static model, each time producing an updated solution. In practice this approach is feasible only if reliable quality solutions of the static model are obtained within a few seconds since the GPS can continuously provide new input regarding the vehicle's positioning and its destinations. We propose a 0–1 programming model to compute exact solutions, together with a variable neighborhood search-based heuristic to obtain approximate solutions for larger instances. Computational results on randomly generated instances are provided to evaluate the performance of the proposed approaches.

Keywords Parking allocation · 0–1 Programming · Variable neighborhood search

1 Introduction

Drivers equipped with a Global Positioning System (GPS) device usually enter their final destination into their device. However, they rarely park their vehicles exactly

✉ Marko Mladenović
marko.mladenovic@univ-valenciennes.fr

1 UVHC, LAMIH UMR CNRS 8201, Mont Houy, 59313 Valenciennes, France

2 HEC Montréal, 3000, chemin de la Côte-Sainte-Catherine, Montreal H3T 2A7, Canada

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

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. Shoup 2006, 1997; Gantelet and Lefauconnier 2006; Caicedo et al. 2016; Davis et al. 2010) further confirm the negative impact of massive unorganized (random) search for parking lots in urban areas.

Research motivation One of the first authors who drew attention to the consequences of unorganized parking was Donald Shoup. In one of his studies (Shoup 2006) he revealed that the search for curb vacant parking slots, even thought they may be cheaper, does not pay off, because other criteria should be taken into consideration, such as the time spent searching 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. In his other paper, (Shoup 1997), he 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. These two papers based their observations on data collected from the USA. Another study by Gantelet and Lefauconnier (2006), based on European insights, reveals that drivers have a tendency to enlarge their search radius. For example, below 15 min, the average distance to the destination is less than 200 m. When the search time exceeds 15 min, the distance becomes more and more significant and can extend beyond 500 m (550 m on average in Lyon for a searching time of 45 min). The authors conclude that searching for parking spaces causes traffic congestion: between 5 and 10% of the traffic in cities and up to 60% on small streets.

Existing practical solutions Most cities have introduced some strategies to tackle this problem. One of the most frequent is the Parking Guidance and Information (PGI), which is displayed on roads and continuously updates neighboring parking availability. Thus, some indoor parking zones include adaptive lighting sensors, as well as parking space led indicators and Indoor Positioning System (IPS).

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[1] 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 desires, he can reserve a place at that parking via this application. An example of a successful project is the ZenPark mobile application,[2] 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.[3] Among

[1] http://www.parkopedia.com.

[2] https://zenpark.com.

[3] http://www.lillemetropole.fr/en/mel.html.

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

other options the users can check the availability of most parking spaces (also public garages) in the city. However, guiding options are not included.

More insights City authorities have, each in its own way, struggled with the consequences of massive unorganized search for available parking. Several studies show that in most cases there are sufficient parking slots for all vehicles and are concerned with the negative environmental impact of constructing more parking (Caicedo et al. 2016; Davis et al. 2010). Therefore, we focus our attention on their allocation to existing facilities in order to avoid traffic jams, reduce travel time, and so on. Furthermore, due to the availability of free slots data in public parking and the results presented in Shoup (1997), in this study we only consider public parking and not curb lots (street parking). Moreover, a GPS signal is available thought most modern devices with 3G/4G connections and the vehicles can exchange information with the central intelligence (server).

Related work The solutions mentioned in the previous paragraph target a single vehicle and allocate to it the parking that suits it best, or serves as a general guideline for currently available parking slots. However, some studies also include other vehicles in the spatial and temporal vicinity and endeavor to allocate the "globally" best parking spot to each vehicle. For example, in Delot et al. (2013) the authors 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 is going to meet a certain event, (Cenerario et al. 2008), and shares the available information according to the encounter probability parameter. In a more recent paper, (Toutouh and Alba 2016), the other aspects of sharing data in decentralized vehicular networks, such as congestion and hazardous road situations are investigated.

A significant portion of articles addressing the parking issue is devoted to the problem of parking pricing. Other studies focus on Electric Vehicles (EVs) and on their specific parking needs. Some authors, such as Teodorović and Lučić (2006) and Delot et al. (2009), focus on reserving parking slots for only one vehicle at a time. Since the reservation of a parking slot is not applicable for most parking needs, we will not consider these papers. In our study, we exclusively focus on the most generic vehicle parking allocation, thus the following paragraph considers the papers that are dealing with the allocation of parking lots to a potentially very large set a vehicles.

So far many researchers have addressed the parking allocation problem. However, there are no standardized mathematical programming models for it, be they deterministic or stochastic. 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 propose various ways of defining the problem at hand. For example, Ayala et al. (2012) opt for a game-theoretic approach and model the parking allocation problem in a similar way as the stable marriage problem, and name it the parking slot assignment game problem. In Verroios et al. (2011), propose a Traveling Salesman Problem (TSP) variant model—the time-varying TSP. The authors also propose a number of algorithms to tackle the proposed model and group vehicles into clusters in order to improve algorithm efficiency. Recently, Roca-Riu et al. (2015)

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

proposed a mathematical programming model for the Parking Assignment Problem (PAP) for delivery vehicles in urban distribution. The authors consider the limited availability of parking slots in urban areas for goods delivery. They model this problem as a variant of the vehicle routing problem with time windows, because it can be assumed that vehicles have to arrive at their destination at a predefined time in order to satisfy the demand. The model proposed in Abidi et al. (2016) has the most resemblance with the model we develop in this paper. The authors allocate vehicles by taking into consideration the fact that different parking have different maximal parking time and propose an efficient heuristic.

All previously mentioned articles consider in the objective function the distance (time) as the main optimization criterion. Furthermore, Verroios et al. (2011) and Delot et al. (2013) consider decentralized networks, i.e. networks in which information is partially accessible by vehicles within a certain radius (see Ilarri et al. 2015 for a detailed survey). Other papers consider that all the information is available to the administrator (centralized system), which can then be used to propose parking lots to vehicles. Several articles advocate the use of GPS data as input for parking guidance (e.g. Gahlan et al. 2016; Mendez et al. 2006). However, to the best of our capabilities, we could not find any contribution with a mathematical programming model.

Contributions In this paper, we consider a potentially very large set of n vehicles, dispersed over a given area. The arrival time t'_{ij} $(i = 1, \ldots, n;\ j = 1, \ldots, m)$ to m potential parking zones can be provided by the GPS. The GPS can also compute the walking time t''_{ij} from each parking to the drivers final destination. These data were used to formulate a 0–1 integer programming model that allocates vehicles to parking lots, by optimizing total travel time (from current location to parking and from parking to destination) of all vehicles. The model is completed with the basic and necessary constraints for any parking assignment problem: capacity and allocation constraints. It can be regarded as a variant of the Generalized Assignment Problem (GAP), and as such is NP-hard in the general case. Our main contributions are the following:

1. a new parking allocation LP model for a set of n connected vehicles, together with a discussion on how to include more realistic constraints for the static PAP;
2. a complexity analysis of the proposed models; for example, it is shown that min–sum type model possesses the integrality property, and therefore is polynomial. However, the min–max static PAP is shown to be NP-hard;
3. a heuristic based on Variable Neighborhood Search (VNS) for it;
4. a discussion on how to extend the model to the dynamic case is provided; in fact we propose to iteratively rerun the static model, since it appears to provide results very fast;
5. an extensive computational analysis of exact and heuristic methods is provided.

Outline The remainder of this paper is organized as follows. Section 2 introduces both combinatorial and mathematical programming models; Sect. 3 presents a VNS-based heuristic for solving it. Section 4 offers comparative results between randomly parked vehicles and the proposed model, solved both exactly and heuristically. We close the paper with concluding remarks in Sect. 5.

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

2 Problem formulation

We first present a combinatorial formulation of the static Parking Allocation Problem (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.

2.1 Combinatorial formulation

Assume that n connected vehicles, equipped with a GPS device, are searching for parking slots in an urban area at time t_0. Also assume that there are m parkings j, each with a known total capacity q_j, $j = 1, \ldots, m$. Once all drivers enter their final destinations, we are then able to determine two types of estimated times or distances (matrices):

- t'_{ij} : estimated time needed by vehicle i to reach parking j, $i = 1, \ldots, n$; $j = 1, \ldots, m$;
- 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 v_{jt} at parking j, for each time t, $t = 1, \ldots, T_j$, where $T_j = \max_i \{t'_{ij}\}$. Note that time $t = 1$ corresponds to t_0 (see Fig. 1).

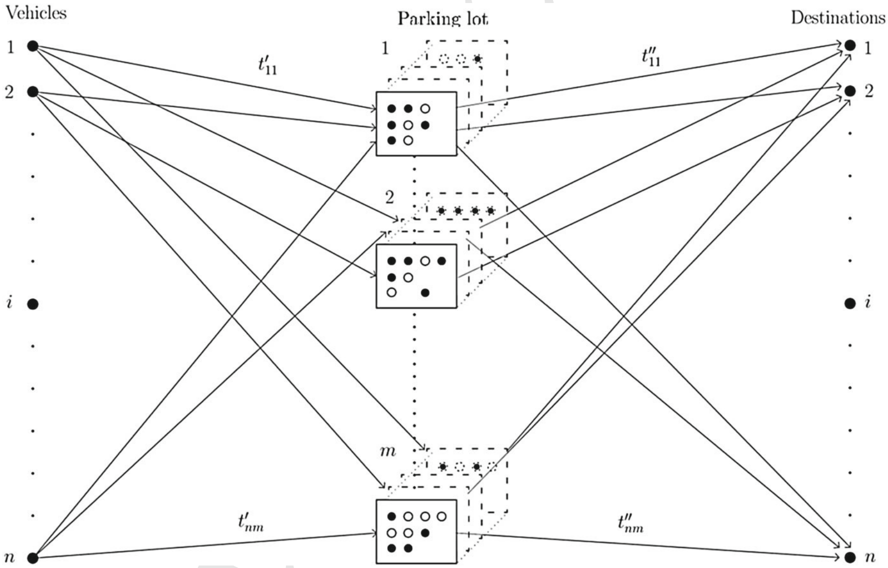

Fig. 1 Graphical representation of the PAP; black circles in the parking column represent the occupied slots, while dotted rectangles represent different times t, and the dotted circles the occupied (available) slots at these time moments

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

Objective function Let $x(i)$ represent the index of the parking to which vehicle i is allocated, and let $\mathcal{P}$ 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:

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

Feasibility Denote by b_j the number of used slots at parking j in the current solution x, and by $u_{j,t}$ 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 gives 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:*

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

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

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 $\mathcal{P}$ 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;

$u_{jt} \leq v_{jt}$: the number of vehicles parked at time t at parking j should be less than or equal to the maximum allowed number v_{jt}.

Estimating the number of free parking lots over time We assume that the v_{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 5 min) 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 v_{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.

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

Dummy parking lot An obvious way to avoid infeasible solutions is to introduce a dummy parking lot $j = 0$. It should have a large capacity, and be very far, i.e. arrival times $t'_{i,0}$ are very large for all vehicles i. So whenever vehicle i cannot be parked at any parking lot $j = 1, \ldots, m$, it will be allocated to the dummy lot $j = 0$.
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 paper, if we refer to a solution as infeasible, this means that at least one vehicle is assigned to the dummy lot.

2.2 Mathematical programming model

It is clear that the principal purpose is to allocate the best parking j to each vehicle i, minimizing the total traveling time. We introduce the binary variable x_{ij} equal to 1 if and only if such an allocation is made. The objective is to minimize the total time traveled:

$$\text{minimize} \sum_{i=1}^{n} \sum_{j=0}^{m} [t'_{ij} + t''_{ij}] x_{ij} \tag{2}$$

subject to

$$\sum_{j=0}^{m} x_{ij} = 1, \qquad (i = 1, \ldots, n) \tag{3}$$

$$\sum_{i=1}^{n} x_{ij} \leq q_j, \qquad (j = 0, \ldots, m) \tag{4}$$

$$\sum_{i=1}^{n} \alpha_{ijt}\, x_{ij} \leq v_{jt}, \quad (j = 0, \ldots, m,\ t = 1, \ldots, T_j) \tag{5}$$

$$x_{ij} \in \{0, 1\}, \qquad (i = 1, \ldots, n,\ j = 1, \ldots, m) \tag{6}$$

where

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

Constraints 3 require that every vehicle be parked, while constraints 4 ensure that the number of vehicles allocated to parking j does not exceed parking capacity q_j. Constraints 5 guarantee that the capacity at each period t for each parking lot j will be respected.

We introduce an additional parking lot $j = 0$ with a large capacity, $q_0 = n$ for example, with sufficient slots at every future time step $v_{0,t} = n, \forall t$, and with larger arrival times $t'_{i,0} > M$, for all i and for some M. If a feasible solution exists, then the dummy parking will remain empty. Otherwise, some drivers would remain without a parking slot and would be temporarily rejected. Note again that only the allocation

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

constraint (3) and the objective function (2) are affected by the dummy facility $j = 0$, since the other constraints are always met.

Properties of the static PAP model The static PAP may be presented as weighted bipartite graph with two types of vertices: vehicle vertices $i = 1, \ldots, n$ and parking vertices $j = 1, \ldots, m$, having weights $w_{ij} = t'_{ij} + t''_{ij}$. We will now prove the property that makes the Boolean model (2)–(6) easy to solve.

Property 3 *The integer programming relaxation of the Boolean model (2)–(6) has integer solutions* $x_{ij} \in \{0, 1\},\ i = 1, \ldots, n;\ j = 1, \ldots, m$.

Proof Let A' be the matrix defined by constraints (3) and (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. Schrijver 1986), the problem defined by (2)–(4) and (6) has the integrality property. This means that the Linear Programming (LP) solution (2)–(4) and $0 \leq x_j \leq 1$ is equivalent to the integer solution of the problem (2)–(4) and (6). In addition, if A' is TU then, $[A'|I]^T$ is also unimodular (Geoffrion 1974). Since the matrix A'' defined by constraints (5) may be transformed into an identity matrix by permuting its rows, we conclude that the matrix defined by (2)–(5) is TU and thus possesses the integrality property. □

Possible extensions of the static PAP From an integer programming standpoint, the basic mathematical programming model (2)–(6) is easy to solve. Here we discuss some possible extensions of the basic model.

- 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. Probably a more realistic and fairer 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:

$$\text{minimize } f(x) = \max_{i=1,\dots,n} \sum_{j=1}^{m} \left[t'_{ij} + t''_{ij}\right] x_{ij} \tag{7}$$

or

$$\text{minimize } f(x, z) = z \tag{8}$$

subject to

$$\sum_{j=1}^{m} \left[t'_{ij} + t''_{ij}\right] x_{ij} \leq z \quad (i = 1, \dots, n), \tag{9}$$

and constraints (3)–(6). Note that, the min–max–max formulation

$$\text{minimize } g(x) = \max_{i=1,\dots,n} \max_{j=1,\dots,m} \left[t'_{ij} + t''_{ij}\right] x_{ij} \tag{10}$$

would yield the same solution as the min–max–sum formulation due to constraints (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 (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.

2.3 Upgrade to a dynamic model

Since the real-world problem is not static, our basic idea—to include time into consideration, consists of repeatedly running the static model, e.g., once every predefined time step (e.g., 1 min). By doing so, we can avoid many unpredictable situations that a static model cannot easily incorporate: (1) a driver already allocated to a parking finds free curb parking; (2) the driver decides to change his destination; (3) the GPS device stops functioning (loses signal) in some vehicles; (4) the time during which the vehicle stays at a parking lot is unpredictable, and using queuing theory in this case would be too unprecise and noisy; (5) vehicles outside of the system can occupy a previously allocated parking slot.

An elegant way to cover many such unpredictable (random) circumstances is simply to solve the problem with the new current input. In 1 min some vehicles will reach the parking they were assigned to, while others will not. In the new solution, most vehicles will keep the same final parking as in the previous solution, but it can happen that some will be reallocated to other parking lots. This is why we need to have a high-speed solution method capable of handling the dynamic nature of the problem by providing solutions of the static model more frequently, since solving the new problem cannot start before the current problem has not been solved. This way, all vehicles appearing in the input are treated with equal priority. So for example, vehicles that would have been left without a parking lot with the current input would be reinserted in the next iteration, along with all other vehicles.

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

3 Variable neighborhood search for parking allocation problem

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. A survey paper on the VNS 0–1 MIP heuristic framework can be found in Hanafi et al. (2015).

Another strong argument for developing a heuristic is the fact that in big cities there could be more than 100,000 vehicles on the streets looking for a parking place. In such cases, 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 any local search heuristic could provide good quality solutions.

Solution representation We present our solution as an array, 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 1.

In line 3, 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 time complexity of the Greedy add algorithm is* $O(nm \log m)$.

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

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

Algorithm 1 Greedy Add(f_{cur}, x, b, u, o, f_v)

1: **procedure** GREEDY_ADD
2: $u \leftarrow v \ \ (u(j,t) \leftarrow v(j,t),\ \forall j, t)$
3: Get order matrix $O_{n\times m} = o(i, j)$
4: Get order $p(i)$ of vehicles $o(i,1)$, $(o(p(1),1) \leq o(p(2),1)\dots)$
5: $b(j) \leftarrow 0, \forall j;\ f_v(i) \leftarrow 0, \forall i;\ f_{cur} \leftarrow 0;\ tt \leftarrow 0$
6: **for** ii = 1 **to** n **do**
7: $i \leftarrow p(ii)$;
8: $tt \leftarrow tt + 1$;
9: **if** $(tt > m)$ **then** 'No feasible solution' **stop**
10: $x(i) \leftarrow o(i, tt);\ j \leftarrow x(i);\ t \leftarrow t'(i, j)$
11: **if** $(u(j,t) = 0$ **or** $b(j) + 1 > q_j)$ **goto** 8
12: $f_v(i) \leftarrow t + t''(i, j);\ f_{cur} \leftarrow f_{cur} + f_v(i)$;
13: $u(j,t) \leftarrow u(j,t) - 1;\ b(j) \leftarrow b(j) + 1;\ tt \leftarrow 0$
14: **end for**
15: **end procedure**

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* GREEDY_ADD *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 $t'_{i,0}$, $\forall i$, 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.

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

Neighborhood structures Obviously, there can be several neighborhood structures for this combinatorial optimization problem. Since our heuristic should be fast, in this paper 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 busy. 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.

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 local search 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 $\mathcal{N}_1^{all}(x)$ is $n \times m$. However, we can significantly reduce it in the following way: reallocate vehicles just to r_v (a parameter) their closest parking $(r_v < m)$.

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: we thus obtain 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_Add` algorithm. A detailed description of our local search is provided in Algorithm 2.

The input variables in `Reallocate_LS`, beside 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

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

Algorithm 2 Reallocate LS($x, f_{cur}, f_v, b, o, r, u, first$)

```
1: procedure REALLOCATE_LS
2:    improve ← true
3:    while improve do
4:       improve ← false
5:       for i = 1 to n do
6:          jj ← x(i);  tt ← t'(i, jj);  f_new ← f_cur − f_v(i)
7:          for j = o(i, 1) to o(i, r) do
8:             t ← t'(i, j);
9:             if (u(j, t) > 0  &  b(j) + 1 ≤ q_j) then
10:               f_new ← f_new + t + t”(i, j)
11:               if f_new < f_cur then
12:                  f_cur ← f_new;  improve ← true
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) − 1
15:                  b(jj) ← b(jj) − 1; u(jj, tt) ← u(jj, tt) + 1
16:                  if (first) return
17:               end if
18:            end if
19:         end for
20:      end for
21:   end while
22: end procedure
```

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 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 `Reallocate_LS`. 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 3.

Note that in `Interchange_LS` we have two vehicles (i_1 and i_2) and two corresponding parking (j_1 and j_2), but 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 $\leq q_j$ here, since vehicles just exchange their parking lots. However, it can happen that at

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

Algorithm 3 Interchange LS($i, j, t, jj, x, t'', tt, b, u, f_v, first$)

```
1: procedure INTERCHANGE_LS
2:    improve ← true
3:    while (improve) do
4:       improve ← false
5:       for i1 = 1 to n − 1 do
6:          j1 ← x(i1);  t1 ← t'(i1, j1)
7:          for i2 = i1 + 1 to n do
8:             j2 ← x(i2)
9:             if j1 ≠ j2 then
10:               t2 ← t'(i2, j2);  t3 ← t'(i1, j2);  t4 ← t'(i2, j1)
11:               if (u(j2, t3) > 0  &  u(j1, t4) > 0) then
12:                  f_new ← f_cur − f_v(i2) − f_v(i1)
13:                  f_v1 ← t'(i1, j2) + t''(i1, j2);   f_v2 ← t'(i2, j1) + t''(i2, j1)
14:                  f_new ← f_new + f_v1 + f_v2
15:                  if f_new < f_cur then
16:                     f_cur ← f_new;  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:                     f_v(j1) ← f_v1;  f_v(i2) ← f_v2
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
```

time t_3 or t_4 there will be no parking place. 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 `Interchange_LS` time complexity of, 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 facts 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. 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

Journal: **10732** Article No.: **9364** ☐ TYPESET ☐ DISK ☐ LE ☐ CP Disp.:**2017/12/23** Pages: **23** Layout: **Small-X**

words, the first time interchange of 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 VND heuristic is used as a local search mechanism. This VNS variant is known as General VNS (GVNS). The basic loop contains the following tree steps: Shaking, VND local search and Neighborhood change. Since the VNS algorithm is well known, we will not describe it here (see Hansen et al. 2016 for a recent survey).

4 Computational results

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.

4.1 Random test instances

We have tested our model and the VNS-based heuristics on randomly generated test instances. We tried to cover real-world situations as well as possible. The number of vehicles n varies from 1000 to 90,000, while the number m of parkings 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 drivers' positions and their destinations are generated according to a discrete uniform distribution in the square $S = [0, 200] \times [0, 200] \in \mathcal{R}^2$. 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 $t'(i, j)$ distances. The distances between parking and destinations $t''(i, j)$ are computed in the same way. The values of matrix $V = (v_{jt})$ 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 v_{jt} for $t = 1, \ldots, T$ (where $T = \max_{i=1,\ldots,n} \max_{j=1,\ldots,m}\{t'_{ij}\}$):

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

In other words, we do not allow the change in the number of free parking slots to be greater than 3, for all parkings 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 1) and were the input does not produce feasible solutions (Table 2). We then switch to larger scale

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

instances, where the number of vehicles searching for a parking lot ranges from 10,000 to 90,000.[4]

4.2 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 1;
- `SeqVND` : sequential VND-based local search, as given in Sect. 3;
- `GVNS` : general VNS, running maximally 10 additional seconds.

Average results on 10 instances, for different pairs of n and m are presented at Table 1.

The third column of Table 1 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 1. The best method is obviously the exact algorithm `CPLEX`. This is expected, since we intentionally propose the basic static PAP model to be fast and "integer friendly". The results obtained by `SeqVND` local search, initialized by `Greedy_add`, 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 10 s. The solutions provided by Greedy are obtained very fast, i.e., it never takes it more than 0.1 s. 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.

[4] The datasets are available on https://goo.gl/H3Nu5H.

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

Table 1 Average results on ten instances for each n and m

Parameters		Exact	% Error			Running time (s)		
n	m	CPLEX	Greedy	seqVND	GVNS	CPLEX	SeqVND	GVNS
1000	10	158,203	4.09	0.10	0.08	0.90	0.33	4.71
	20	147,250	4.43	0.16	0.14	1.21	0.55	6.43
	30	144,064	4.66	0.22	0.19	1.55	0.60	7.40
3000	10	507,136	6.28	0.08	0.08	1.98	4.41	3.13
	20	451,402	4.29	0.14	0.13	2.88	6.15	2.92
	30	432,211	4.95	0.15	0.14	4.28	8.97	4.26
5000	10	822,996	5.78	0.07	0.06	2.64	19.12	1.58
	20	728,954	3.22	0.10	0.10	4.99	29.76	2.62
	30	729,491	5.51	0.12	0.12	7.76	54.06	1.73
7000	10	1,131,207	4.94	0.22	0.22	3.62	45.75	0.85
	20	1,024,958	3.81	0.13	0.13	6.54	82.28	2.27
	30	1,005,248	4.01	0.12	0.12	8.23	133.69	0.00
9000	10	1,453,969	5.86	0.05	0.05	4.61	75.96	1.24
	20	1,329,617	4.75	0.09	0.09	9.20	120.09	0.85
	30	1,286,264	3.92	0.12	0.12	10.90	161.47	0.00

4.3 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 v per time step t. Tests are conducted on four instances for each n and $m = 50$. The running time of the Reduced VNS is fixed to 5 s, 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 place (see Property 5).

The results are reported in Table 2. 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 six to eight give the 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 $(f_{heur} - f_{exact})/f_{exact} \times 100$.

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 instances, there is not enough time to reach a higher precision.

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

Table 2 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

n	# of unparked	Objective values			Running time (s)			% error	
		CPLEX	Greedy	RVNS	CPLEX	Greedy	RVNS	Greedy	RVNS
1000	3	706,592	733,584	714,842	0.58	0.02	5.00	3.82	1.17
	46	879,492	964,844	894,230	0.56	0.00	5.00	9.70	1.68
	15	743,327	800,079	759,107	0.52	0.00	5.00	7.63	2.12
	18	750,107	832,561	760,679	0.61	0.00	5.00	10.99	1.41
Average	20.50	769,879.5	832,767.0	782,214.5	0.57	0.00	5.00	8.04	1.59
3000	254	287,6571	3,138,545	2,900,863	2.19	0.02	5.00	9.11	0.84
	275	2,969,645	3,299,489	2,999,241	2.40	0.02	5.00	11.11	1.00
	64	2,230,716	2,343,062	2,253,236	1.98	0.02	5.00	5.04	1.01
	176	2,592,602	2,780,696	2,614,336	1.97	0.02	5.00	7.26	0.84
Average	192.25	2,667,383.5	2,890,448.0	2,691,919.0	2.13	0.02	5.00	8.13	0.92
5000	497	5,058,142	5,434,074	5,142,612	4.74	0.00	5.00	7.43	1.67
	564	5,312,057	5,713,841	5,410,361	4.87	0.00	5.00	7.56	1.85
	63	3,629,276	3,884,942	3,712,448	4.35	0.00	5.00	7.04	2.29
	582	5,357,472	5,862,696	5,493,064	4.61	0.02	5.00	9.43	2.53
Average	426.50	4,839,237.0	5,223,888.0	4,939,621.0	4.64	0.00	5.00	7.87	2.09
7000	757	7,279,344	7,885,014	7,505,630	7.78	0.02	5.00	8.32	3.11
	559	6,592,231	7,149,541	6,827,215	7.05	0.02	5.00	8.45	3.56
	1003	8,058,449	8,794,531	8,329,667	8.35	0.02	5.00	9.13	3.37
	773	7,518,119	8,160,985	7,793,777	7.81	0.02	5.00	8.55	3.67
Average	773.00	7,362,036.0	7,997,518.0	7,614,072.0	7.75	0.02	5.00	8.61	3.43

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

Table 2 continued

n	# of unparked	Objective values			Running time (s)			% error	
		CPLEX	Greedy	RVNS	CPLEX	Greedy	RVNS	Greedy	RVNS
9000	1549	11,283,644	12,046,312	11,627,640	10.32	0.03	5.00	6.76	3.05
	604	7,994,571	8,535,257	8,237,163	8.43	0.03	5.00	6.76	3.03
	1164	9,832,471	10,622,843	10,136,153	8.84	0.03	5.00	8.04	3.09
	294	7,346,493	8,177,867	7,854,405	14.50	0.03	5.00	11.32	6.91
Average	902.75	9,114,295.0	9,845,570.0	9,463,840.0	10.52	0.03	5.00	8.22	4.02

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

Table 3 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

n	# of unparked	Objective values			Running time (s)			% error	
		CPLEX	Greedy	RVNS	CPLEX	Greedy	RVNS	Greedy	RVNS
10,000	1104	10,338,915	11,212,253	10,749,531	10.67	0.03	5.00	8.45	3.97
	796	9,359,846	10,220,204	9,887,008	13.00	0.02	5.00	9.19	5.63
	313	7,868,790	8,327,986	8,176,778	12.16	0.03	5.00	5.84	3.91
	1027	10,162,548	11,060,328	10,695,966	12.48	0.03	5.00	8.83	5.25
Average	810.00	9,432,524.0	10,205,192.0	9,877,321.0	12.08	0.03	5.00	8.08	4.69
30,000	1642	25,688,032	27,225,734	27,139,640	61.82	0.08	5.00	5.99	5.65
	3	20,484,031	20,623,065	20,619,435	29.74	0.06	5.00	0.68	0.66
	1521	25,558,158	27,832,720	27,731,256	78.90	0.08	5.00	8.90	8.50
	1425	24,874,247	26,329,169	26,267,431	53.96	0.08	5.00	5.85	5.60
Average	1147.75	24,151,118.0	25,502,672.0	25,439,442.0	56.10	0.08	5.00	5.35	5.10
50,000	7569	60,307,988	66,659,792	66,421,052	252.50	0.14	5.00	10.53	10.14
	1362	38,926,503	41,748,795	41,733,403	179.12	0.11	5.00	7.25	7.21
	30	34,697,447	36,491,253	36,470,753	152.78	0.11	5.00	5.17	5.11
	2379	42,713,609	46,134,033	46,100,037	227.07	0.14	5.00	8.01	7.93
Average	2835.00	44,161,388.0	47,758,468.0	47,681,312.0	202.87	0.13	5.00	7.74	7.60
70,000	7062	n/m	7,562,6571	75,552,711	–	0.19	5.00	–	–
	6214	n/m	74,843,077	74,762,329	–	0.19	5.00	–	–
	3822	n/m	65,062,124	64,988,624	–	0.17	5.00	–	–
	6491	n/m	73,271,387	73,220,405	–	0.19	5.00	–	–

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

Table 3 continued

n	# of unparked	Objective values			Running time (s)			% error	
		CPLEX	Greedy	RVNS	CPLEX	Greedy	RVNS	Greedy	RVNS
Average	5897.25	–	72,200,792.0	72,131,016.0	–	0.19	5.00	–	–
90,000	8618	n/m	97,054,560	96,997,926	–	0.25	5.00	–	–
	474	n/m	66,596,853	66,589,769	–	0.23	5.00	–	–
	9688	n/m	102,440,048	102,380,832	–	0.25	5.00	–	–
	1079	n/m	68,548,146	68,540,540	–	0.23	5.00	–	–
Average	4964.75	–	83,659,904.0	83,627,264.0	–	0.24	5.00	–	–

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

4.4 Infeasible large instances

We also compared exact and heuristic methods on instances with $n = 10{,}000$, 30,000, 50,000, 70,000 and 90,000, 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 [1000×1000] and the location of dummy facility is set at the point with coordinates (1700, 1700). Since the solution should be obtained within less than 5 s, among several VNS variants, we run only Reduced VNS after the Greedy initial solution (Table 3).

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 s. Moreover, for more than 70 thousand vehicles, our PC ran out of memory (16 GB). 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.

5 Conclusions

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 their mostly non-deterministic input. In this paper, we have proposed a new mathematical programming model that uses 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 (SPAP). We showed that 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 $n \geq 30{,}000$), 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 5 s, for example. Therefore, it is more important to compute an approximate solution fast within a fixed time limit, rather than 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.

Future work may follow the following directions: (i) to test our models on real parking data, including more elaborate dynamic variants; (ii) to develop a VNS heuristic and exact methods for the min–max–sum variant of PAP; (iii) to develop exact solution procedures for SPAP that would use more the problem specific knowledge and not be based on commercial solvers. In other words, trying to build a strictly polynomial exact method for SPAP, in order to reduce the time of the exact solution method.

Journal: 10732 Article No.: 9364 TYPESET DISK LE CP Disp.:2017/12/23 Pages: 23 Layout: Small-X

References

Abidi, S., Krichen, S., Alba, E., Bravo, J.M.M.: A hybrid heuristic for solving a parking slot assignment problem for groups of drivers. Int. J. Intell. Transp. Syst. Res. **15**(2), 85–97 (2016)

Ayala, D., Wolfson, O., Xu, B., DasGupta, B., Lin, J.: Parking in competitive settings: a gravitational approach. In: 2012 IEEE 13th International Conference on Mobile Data Management, IEEE, pp. 27–32 (2012)

Caicedo, F., Lopez-Ospina, H., Pablo-Malagrida, R.: Environmental repercussions of parking demand management strategies using a constrained logit model. Transp. Res. Part D Transp. Environ. **48**, 125–140 (2016)

Cenerario, N., Delot, T., Ilarri, S.: Dissemination of information in inter-vehicle ad hoc networks. In: 2008 IEEE Intelligent Vehicles Symposium, IEEE, pp. 763–768 (2008)

Davis, A.Y., Pijanowski, B.C., Robinson, K.D., Kidwell, P.B.: Estimating parking lot footprints in the Upper Great Lakes Region of the USA. Landsc. Urban Plan. **96**(2), 68–77 (2010)

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

Delot, T., Ilarri, S., Lecomte, S., Cenerario, N.: Sharing with caution: managing parking spaces in vehicular networks. Mob. Inf. Syst. **9**(1), 69–98 (2013)

Gahlan, M., Malik, V., Kaushik, D.: GPS based parking system. Compusoft **5**(1), 2053–2056 (2016)

Gantelet, E., Lefauconnier, A.: The time looking for a parking space: strategies, associated nuisances and stakes of parking management in France. In: Association for European Transport, Europe Transport Conference 2006, Strasbourg, France (2006)

Geoffrion, A.: Lagrangean relaxation for integer programming. In: Approaches to Integer Programming, No. 2 in Mathematical Programming Studies, Springer, Berlin, pp. 82–114 (1974)

Hanafi, S., Lazić, J., Mladenović, N., Wilbaut, C., Crévits, I.: New variable neighbourhood search based 0–1 MIP heuristics. Yugosl. J. Oper. Res. **25**(3), 343–360 (2015)

Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S.: Variable neighborhood search: basics and variants. EURO J. Comput. Optim. **5**(3), 423–454 (2016)

Ilarri, S., Delot, T., Trillo-Lado, R.: A data management perspective on vehicular networks. IEEE Commun. Surv. Tutor. **17**(4), 2420–2460 (2015)

Mendez, G., Herrero, P., Valladares, R.: SIAPAS: a case study on the use of a GPS-based parking system. In: On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, Springer, vol. 4277, pp. 945–954 (2006)

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

Schrijver, A.: Theory of Linear and Integer Programing. Wiley, New York (1986)

Shoup, D.C.: High cost of free parking. J. Plan. Educ. Res. **17**(1), 3–22 (1997)

Shoup, D.C.: Cruising for parking. Transp. Policy **13**(6), 479–486 (2006)

Teodorović, D., Lučić, P.: Intelligent parking systems. Eur. J. Oper. Res. **175**(3), 1666–1681 (2006)

Toutouh, J., Alba, E.: 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, pp. 433–442 (2016)

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

Journal: **10732** Article No.: **9364** TYPESET DISK LE CP Disp.:**2017/12/23** Pages: **23** Layout: **Small-X**

Journal: 10732
Article: 9364

Author Query Form

Please ensure you fill out your response to the queries raised below and return this form along with your corrections

Dear Author

During the process of typesetting your article, the following queries have arisen. Please check your typeset proof carefully against the queries listed below and mark the necessary changes either directly on the proof/online grid or in the 'Author's response' area provided below

Query	Details required	Author's response
1.	Please confirm if the corresponding author is correctly identified. Amend if necessary.	
2.	Kindly check and confirm the institute name is correctly identified for affiliation 1, and amend if necessary.	
3.	Please confirm the direct and indirect reference citations are correctly identified.	
4.	Please check and confirm the inserted citation of Table 3 is correct. If not, please suggest an alternative citation. Please note that Tables should be cited in sequential order in the text.	