Automated Planning and Control for High-Density Parking Lots

Pedro M. d'Orey, José Azevedo, Michel Ferreira
Instituto de Telecomunicações, Universidade do Porto
Rua do Campo Alegre, 1021/1055
4169-007 Porto, Portugal
{pedro.dorey, jose.azevedo, michel}@dcc.fc.up.pt

Abstract

Parking is a key component for efficient urban mobility with implications on congestion and the urban landscape. We propose planning strategies for automated, high-density parking lot systems that efficiently execute (i) selection of vehicle destination and (ii) conflict-free motion planning for vehicle input and output operations. The practical performance of the system is demonstrated through simulation using an empirical dataset. The system allows halving the space requirements for parking vehicles while providing fast access to vehicles and keeping low the in-park travel distances when comparing with conventional parking lots.

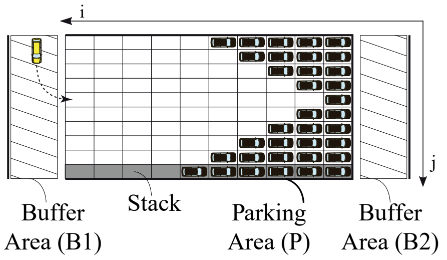


Figure 1: High-density parking lot with n interacting stacks and two buffers areas (B1, B2). Buffer area B2 is optional.

Introduction

Parking causes severe social, economic and environmental problems in major urban areas. Shoup estimated that 30% of the traffic in urban areas is due to *cruising for parking* given the imbalance between on and off-street parking prices (Shoup 2006). Additionally, studies have shown an excessive land use solely dedicated for parking in business districts or city centers (e.g., (Edwards 2012) states that in downtown Atlanta 21% of the land is dedicated for parking). The true cost of owning a vehicle is often overlooked as Shoup mentions: *It is unfair to have cities where parking is free for cars and housing is expensive for people*.

Vehicle manufacturers have presented various systems to automate parking operations. *Self-parking* [e.g., (Wang et al. 2014)] systems automatically perform parallel and perpendicular maneuvers by controlling the vehicle actuators based on sensing data. Automated Valet Parking [e.g., (Conner et al. 2007)] combines self-parking with autonomous driving, allowing the passenger to leave at its destination rather than at the parking lot. Automated robotic parking systems, which resort to electric elevators and rotating/sliding platforms to automatically park vehicles in a high-density parking configuration, allow optimizing parking capacity but present high capital and operational costs.

To reduce space requirements for parking, while simultaneously maintaining low capital and operational costs, (Ferreira et al. 2014a) (Ferreira et al. 2014b) proposed an automated, high-density parking system enabled by drive-by-wireless and vehicular networks[1]. The high-density parking configuration - where inter-vehicle distance is kept to a minimum - improves considerably land use. Instead of using mechanical platforms, this system relies on *collaborative vehicle mobility*, that replaces the concept of a static parking session by a slow motion idle state that is particularly compatible with electric vehicles. A parking lot controller (PLC) governs the collaborative in-park vehicle mobility by instructing selected vehicles to perform maneuvers to allow other vehicles to reach their targets. The PLC is responsible for determining movement plans for (cohorts of) vehicles whose motions might be interdependent due to the parallel task execution and the high-density parking configuration.

We go beyond our previous works by proposing efficient planning strategies in a high-density parking scenario considering the interdependent motions of vehicles arising from the parallel task execution. We solve the task of simultaneously moving (groups of) vehicles in a high-density parking lot between: (i) the parking entrance and the assigned parking position (*storage*), (ii) the current and the destination position (*relocation*) and (iii) the current parking position and the parking exit (*removal*), considering that the path between these two locations has movable obstacles. The strategies also consider selecting the *vehicle destination* in the high-density area depending on a number of criteria (e.g. exit time). The system aims at minimizing the in-park mobility in terms of travel distance and/or vehicle maneuvers, while ensuring low vehicle retrieval times.

[1]A small-scale demonstration is available at http://www.dcc.fc.up.pt/~pedro.dorey/parking.html.

Related Work

The problem of planning collision-free paths for multiple (groups of) vehicles in a high-density parking configuration relates to the *Multi-agent Path Planning Problem*, which has been shown to be a PSPACE-hard problem (Wang, Botea, and others 2008).(Wang and Botea 2009) proposed a multi-agent path planning algorithm for grid maps that runs in low-polynomial time. (Wang and Botea 2011) proposed a Scalable Multi-Agent Path Planning Algorithm with Tractability and Completeness Guarantees for the class of problems termed *Slidable*. (de Wilde, ter Mors, and Witteveen 2013) presented the *Push and Rotate* algorithm to move agents to specific positions through evasive movements of other agents. Similar tasks are found in other application domains, namely goods transfer (Ma et al. 2016b), warehouse storage (Huetter 2016), transshipment (Yu and LaValle 2013), container stacking (Salido et al. 2009), among many others. (Ma et al. 2016a) discussed the challenges arising from the application of multi-agent path finding to real world scenarios.

The problem herein considered has unique features that differentiates it from other similar problems: (i) obstacles (i.e., vehicles) are self-movable, (ii) several vehicles can be moved in a platoon between different stacks, (iii) the selection of the *vehicle destination* is tightly coupled with path planning in high-density parking lots, (iv) vehicles can exchange positioning information and (v) there exists regularity and predictability of certain system parameters (e.g. vehicle exit times), which creates new challenges and opportunities for further optimization. On the other side, there exist strict requirements on the algorithmic execution times since (i) the frequent storage and retrieval of vehicles creates reconfigurations of the system layout in a high-density parking area that can contain hundreds of vehicles and (ii) the system should guarantee short vehicle retrieval times.

Planning and Control Framework

Setting The automated parking lot (PL) is composed by a parking area (P), and one or more buffer areas ($B1$ and $B2$) as depicted in Fig. 1. Vehicles are parked in equal sized cells (s_{ij}) in a grid structure P. We impose restrictions on the geometry of the scenario to reduce the problem complexity. Let $P = \{s_{ij} : 0 < i < m, 0 < j < n\}$ be a rectangular grid composed of $m * n > 1$ parking spaces s_{ij} with center on (x_{ij},y_{ij}), width w_{ij} and height h_{ij}. The grid P can be viewed as being composed of n *interacting* stacks. Let $B = \{b_{ij} : 0 < i < M, 0 < j < N\}$ be a contiguous rectangular area to P with three main functions: (i) transfer area for vehicles moving between different stacks, (ii) temporary storage and (iii) circulation area for entering and exiting vehicles. The buffer area can also be used as transfer area for moving a vehicle within one column.

Let V be a finite set of vehicles where $|V| \leq |P|$ and $I : V \rightarrow P$ be the assignment function of vehicles to parking spaces. We consider that vehicles take unit discrete steps in one of four directions (up, down, left, right) (i.e., $U = \{(0, 1), (0, -1), (1, 0), (-1, 0)\}$) to non-occupied cells ($I^{-1}(s_{i'j'}) = \emptyset$) of the parking lot ($s_{ij} \in P$ or B). The transfer of vehicles between different columns of P (through B) has to meet several dynamic constraints, such as, minimum turning radius R_{min} for all vehicles, i.e., $M_t^B = \{s_{ij} \rightarrow s_{i'j'}\}$ iff $\exists\, a : r_{i \rightarrow i'} \geq R_{min}$, where $r_{i \rightarrow i'}$ is the turning radius from column i to column i'. In a low-density scenario, the routing of vehicles between different stacks can be done solely through the parking area P without resorting to the buffer areas. Vehicles in a given stack may form a cohort to travel simultaneously in the same direction reducing the in-park travel times.

Our main goal is to plan a sequence of actions (i.e., vehicle movements) that change the current state x in state space X over time to allow the entry or exit of vehicles from the high-density parking area. The set of vehicles V is partitioned into active V_A (i.e., entering, leaving or internal transfer) and parked V_P vehicles (i.e., no planned operations).

Assumptions Besides the geometric and motion constraints indicated previously, we consider that : (i) accurate positioning information is available for vehicles, (ii) highly precise remote control of vehicles is possible, (iii) vehicles are able to communicate via V2V/V2I networks, (iv) vehicles can sense obstacles in the environment and (v) selected system parameters (e.g. vehicle exit times) are available or can be inferred with a high accuracy. Note that OEMs have already introduced in premium vehicles, several systems that allow remote vehicle control [cf. (Uhlemann 2015)] and environment sensing (e.g., radar), and vehicular communications will be a reality in the coming years.

General Functioning The high-density parking system demands solving the tasks: (i) selection of vehicle destination (i.e., final parking position), (ii) conflict-free motion planning for vehicle input and vehicle output and (iii) variable vehicle size compacting. In conflict-free path planning it is ensured that the trajectories of vehicles are not overlapping in time and space. Compacting vehicles of variable size in the parking lot relates to the Bin Packing problem (Coffman, Garey, and Johnson 1997), which is a combinatorial NP–hard problem. To reduce the problem complexity, we focus on the first two tasks and consider these tasks as independent subproblems. In the following, we focus in a high-density parking configuration with one buffer area (B1).

The general system functioning is presented in Alg. 1. For each vehicle v in the set of active vehicles V_A, the procedure computes a feasible path $\pi(v)$ to its target position (lines 2-8); the determination of the vehicle target position (line 4) is dependent on the vehicle parking strategy. A procedure (line 9) then heuristically selects the order by which paths are executed (e.g., exiting vehicles can have higher priority than entering vehicles), which can impact the execution speed. Next, one or more paths are executed possibly simultaneously; the execution of these tasks will depend whether the target position is immediately available (lines 16-18) or not (lines 11-15). The plan execution phase considers two basic primitives. In the $push\_move$ primitive, vehicles are moved along a given direction in the stack until a given vehicle reaches its target position. In the $path\_clearing$ primitive vehicle(s) are moved away from their current stack to allow a given vehicle to reach its destination. Vehicles affected by $path\_clearing$ might return to their original positions or be directed to one or several new stacks.

```
Input  : active vehicles $V_A$, state $x$, function $I$, grid $G$
Output: Feasible paths $\pi(v)$ for vehicles in $V_A$
1  $p\_feasible \leftarrow False$;
2  for each $v \in V_A$ do
3      while $p\_feasible$ not found do
4          $s_g^v \leftarrow Vehicle\_destination(s_o^v, I, G)$ ;
5          $\pi(v) \leftarrow Shortest\_path(G, s_o^v, s_g)$ ;
6          $p\_feasible \leftarrow path\_feasibility(p, V_A, I, G)$ ;
7      end
8  end
9  $prio \leftarrow calculate\_priorities()$ ;
10 for each $v \in V_A$ in order do
11     if (goal cell $s_g^v$ is blocked) then
12         $Path\_clearing(\pi(v), v_b)$ for vehicle(s) $v_b$ ;
13         $Push\_move(\pi(v), I)$ of vehicle $v$ ;
14         strat $\leftarrow Relocation()$ ;
15         $Push\_move(\pi(v_b), I)$ of blocking vehicle(s) $v_b$
16     else
17         $Push\_move(\pi(v))$ of vehicle $v$ ;
18     end
19 end
```
Algorithm 1: General functioning

```
Input  : Path $\pi(v) = \{s_o^v, ..., s_g^v\}$, $I$, $V_B$
Output: Move Plan MP
1  MP $\leftarrow$ () ;
2  $\pi(full) \leftarrow \pi(v)$ ;
3  for each $v_b \in V_B$ in order do
4      $\pi(full) \leftarrow \pi(full).append(I(v_b))$
5  end
6  $\pi(full) \leftarrow \pi(full)[|V_B|:]$ ;
7  pos $\leftarrow$ I(v) ;
8  while $I^{-1} = \emptyset$ do
9      MP $\leftarrow$ MP.append(I(v)) ;
10     pos $\leftarrow$ next($\pi(v)$)
11 end
12 dst $\leftarrow \pi(full)$[-1] ;
13 src $\leftarrow$ dst ;
14 while src $\neq s_0^v$ do
15     while $I_0^{-1}(src) = \emptyset$ do
16         src $\leftarrow$ previous($\pi(full)$, src);
17     end
18     t $\leftarrow$ src ;
19     while t $\neq$ dst do
20         MP $\leftarrow$ MP.append(t);
21         t $\leftarrow$ next($\pi(full)$)
22     end
23     dst $\leftarrow$ src
24 end
```
Algorithm 2: $push\_move$ Algorithm

– Selection of Vehicle Destination is executed for vehicles entering the system (*storage*) and for relocating parked vehicles to allow the entry/exit of other vehicles (*relocation*), while ensuring low in-park mobility and fast access to vehicles. The selection of the vehicle destination is done heuristically based on a single or a combination of geometric (e.g., number of vehicles in each of the n stacks), time (e.g., vehicle exit times), distance (e.g., sum of the distance between input-destination and destination-output), among other criteria. Another decision variable to consider is the position within a given column (e.g., in an empty column vehicles can be placed anywhere in between the closest or the furthest position relative to a given reference buffer area). If the selected target cell is occupied, an additional motion planning between the current position and the new destination has to be considered to allow the entry/exit of the vehicle. The motion plan might take two forms: (i) a push move or (ii) path clearing. The output of this subtask is the determination of a single or a sequence of destination cells for push move and path clearing, respectively.

– Conflict-free Motion Planning Storing a new vehicle, removing an in-park vehicle or relocating a vehicle(s) within the system implies finding a plan to clear a destination location $d \in P$ ($push\_move$) or a set of cells $C = \{c \subset P : I^{-1}(c) = \emptyset\}$ ($path\_clearing$). We allow simultaneous motion of vehicles in parallel stacks (irrespective of the direction) considering that (groups of) vehicles need to contend for access to the buffer area if their paths are overlapping in time and space. The simultaneous motions allows decreasing considerably the time required to clear a given cell.

In the push task for *storage*, we create a virtual path between the current location and the destination location s_g^v and move simultaneously the vehicle(s) along the path so that the entering vehicle can reach cell s_g^v while respecting the restrictions on cell occupancy and stack size. Regarding cell occupancy restrictions, only one vehicle can occupy a given cell at a given time and a vehicle can only move to a neighbor cell after this is freed by the preceding vehicle. The $push\_move$ primitive is presented in Alg. 2. First, we construct the path $\pi(full)$ from the cell before the first obstacle of vehicle v to a virtual destination that is $|V_B|$ positions away from the last object in the obstacle ensemble (lines 2-6). Following, we add to the plan the transitions to move vehicle v to a cell adjacent to the first obstacle (lines 8-11). We then initialize the variables dst and src to the last free cell and the last cell on the virtual path (lines 12-17) doing a backward search. Additional actions are added to the plan MP to move the shuttle ensemble from src to dst (lines 18-22) (forward execution). The procedure is repeat with updated src and dst pointers until v has reached its destination (line 14). This primitive is similar to the algorithm proposed by (Huetter 2016) but additional path computations to bring vehicle v close to the obstacle ensemble and with additional restrictions on the number of pushes (since the objective is for vehicle v to reach its destination rather than making one cell available). The cost of a $push\_move$ is solely equal to combined cost of the length of path $\pi(v)$ plus the movements of blocking vehicles in the obstacle ensemble.

Under certain circumstances (e.g., if vehicle ordering needs to be considered), it might be necessary to free a set of cells in P to allow a vehicle to reach its target position. If solely considering one buffer area, path clearing is necessary for vehicle *retrieval*, either for (i) *relocation* of vehicle(s) between stacks or (ii) vehicle *removal* from the parking area. In $path\_clearing$ for *vehicle retrieval*, we first create a virtual path between the current vehicle position o (I(v) or B, respectively) and destination d (park exit or vehicle destination, respectively) followed by path clearing that moves the vehicle(s) out from the current stack to allow the exit/entry of vehicle v (lines 11-13 of Alg. 1). Depending on the relocation strategy, vehicles might be pushed to one or several stacks (lines 14-15 of Alg. 1). The order of the tasks in lines 13 and 15 can be swapped if beneficial in terms of travel distance of execution speed, i.e., vehicles are moved directly to their target positions and not parked temporarily in B or P.

Planning and Control Strategies

In the following, we present two planning strategies for parking vehicles in a high-density configuration that make use of the framework presented in the previous section. The first proposed planning strategy was termed **Smallest Length Stack (SLS)**. The selection of the target parking position is done heuristically. The criteria for selecting the target vehicle position is the current stack occupancy. Stack occupancy is defined as the number of parking spaces that are currently occupied by vehicles. Using the assignment function I of vehicles to parking spaces that defines the current parking lot state, the procedure iteratively calculates the stack occupancy of each stack and updates a pointer to the smallest stack if certain $conditions$ are met (e.g., possible to push vehicles along path to allow entering vehicle to reach designated parking position). If several stacks have the minimum occupancy, the vehicle is placed in the leftmost stack. Note that we consider the row within a given stack as a parameter since our goal is to understand the impact of this parameter on the system performance.

Depending on the current parking configuration I, new vehicle(s) entering the parking lot might need to perform a $push\_move$ to the vehicles(s) in the selected stack if the target position is not immediately reachable (i.e., target position is being occupied by other vehicles). The exit of a vehicle from the parking lot might cause the relocation of other vehicles through a $path_clearing$ if these are blocking the current path to parking exit. In the SLS strategy, vehicles being relocated are assigned a new parking position as if they were now entering the parking lot (i.e., the vehicle destination procedure is applied to each relocating vehicle). Vehicles can potentially be re-directed to N different stacks.

In the **Conditional Order on Arrival (COA)** strategy, the main objective is to have vehicles in stacks ordered by exit time (i.e., we park vehicles with smallest parking duration closer to to the buffer area). In the following, we assume that the exit times of vehicles in the parking lot are known in forehand. Consider that $t^{out}(v)$ and $t^{out}(v^{st}_{first})$ is the exit time of the a vehicle v entering the parking lot and of vehicle in stack st closest to the buffer area, respectively. In order to reduce the probability of moving vehicles to allow the exit of a given car, the vehicle destination strategy parks together vehicles with similar exit times. Thus, the selection of the vehicle destination in COA is done based on the heuristic δ_{st} minimum positive time interval between $t^{out}(v)$ and $t^{out}(v^{st}_{first})$. The procedure iterates over the different rows, selects the vehicle closest to buffer area ($first\_vehicle$) and selects the stack with minimum positive δ_{st}. We solely select the the vehicle closest to the buffer area to reduce the in-park mobility due to the introduction of a new vehicle into the system (i.e., we avoid the execution of $path\_clearing$ during vehicle arrival). If no solution can be found, the vehicle is parked in the stack with smallest occupancy (similar to the vehicle destination procedure of SLS).

In general, the exit of a vehicle from the parking lot does not require the execution of $path\_clearing$ primitives as vehicles are ordered by exit time, reducing the in-park mobility. However, the COA strategy does not guarantee that vehicles are *always* ordered by exit time in the stacks. Thus, whenever an exiting vehicle is blocked by other vehicles, it will trigger the execution of the $path\_clearing$ procedure so that the path to parking lot exit is freed. The COA strategy makes use of the *1-N* vehicle relocation pattern. In this pattern, each vehicle being relocated is assigned a new parking position in the stack with smallest δ_{st}.

Results & Discussion

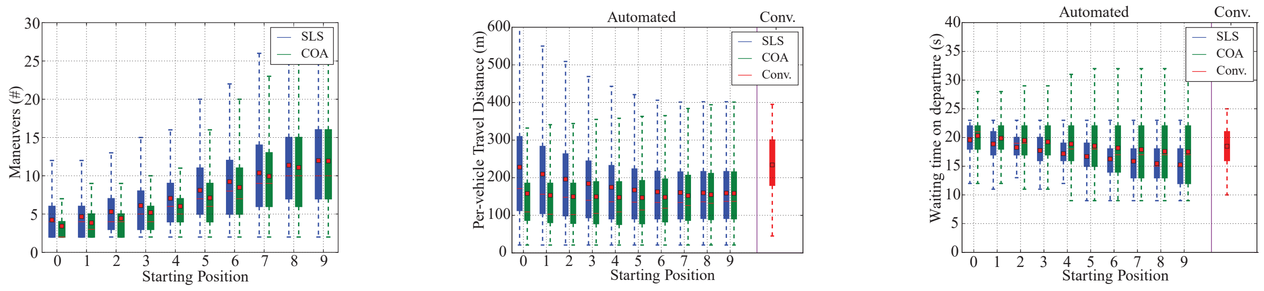
The system performance was evaluated through discrete event simulation resorting to an empirical dataset containing parking entry and exit events. The simulator keeps track of the current parking lot state (e.g., assignment of vehicles to parking spaces, pending action list) and mimics the maneuvers executed by vehicles and their interactions. Vehicles perform kinematically-valid motions. The high-density parking lot can hold up to 100 vehicles (Fig. 1). Note that the simulated parking lot solely considers one buffer area (B1).

Dataset We performed experiments using a dataset of a parking lot in the city of Porto, Portugal, with capacity to hold up to 100 vehicles. The dataset [2] comprises around 100 thousand parking events for the year 2013. Each event consists of the following data: timestamp, gate identifier, event type (IN/OUT) and person identifier. Abnormal parking events (e.g., duration smaller than 10 min and larger than 24 h) have been removed from the dataset. The dataset presents variable parking duration and variable vehicle inter-arrival times (i.e., time interval between two vehicle arrivals) depending the time of the day, day of the week and month. The parking duration follows a bimodal distribution (modes: 4 h and 8.5 h). The average inter-arrival time is 339 $\pm$ 514 s.

Metrics We consider four metrics aggregated per-vehicle:

- **Parking space per vehicle** (m^2): quotient between the total parking area and the parking capacity.
- **Maneuvers** (#): total number of maneuvers performed between each vehicle start until the last full stop for vehicle entry, exit and inter-stack mobility. In conventional parking lots this metric is fixed to two (1 entry + 1 exit).
- **Travel distance (m)**: total distance traveled for vehicle entry/exit procedures and inter-stack mobility.
- **Waiting time on departure (s)**: time elapsed between pickup request and vehicle arrival to parking lot entrance.

[2]The empirical dataset is available at (d'Orey 2017)



(a) Maneuvers (#) (b) Travel distance (m) (c) Waiting time on departure (s)

Figure 2: Per-vehicle Metrics for Automated and Conventional Parking Lots.

Results

Space efficiency The high-density parking lot occupies a total area of 1100 m^2 (11 m^2/vehicle) considering that each parking space has an area of 2 x 5 m^2 and the buffer B1 has an area of 100 m^2. The conventional parking lot has an area of approximately 2250 m^2 (22.5 m^2/vehicle). Thus, the implementation of the system allows decreasing by 50% the effective space occupied per car. The space optimization arises from the compact structure of the self-automated parking lots, the smaller parking spaces (spacing between vehicles can be kept to a minimum in the former) and the absence of several structures, namely circulation lanes for vehicles and pedestrians. The benefits of high-density parking lots could even be higher as (Hill et al. 2005) state that a good static efficiency of a car park with 100 parking spaces at 90° is 22 m^2 per parking space.

Impact of planning strategy Fig. 2 presents the main results. We observe that *COA* has better performance than *SLS* in terms of total travel distance and number of maneuvers due to the more careful vehicle storage planning as vehicles can be ordered by exit time. For both strategies, the number of maneuvers is low for vehicles stored at the top of the stacks (starting position 0) at the cost of additional travel distance for the SLS algorithm. When comparing automated and conventional parking, we observe that for automated parking (i) the number of maneuvers increases due to inter-stack mobility and (ii) the total travel distance decreases for the majority of the vehicles due to the compact high-density configuration. However, the travel distance can increase for a small set of vehicles (specially for SLS) when comparing with conventional parking lots due to the in-park vehicle mobility to allow the entry and exit of vehicles. We also conclude that waiting times on departure are (i) small for all considered planning strategies and (ii) similar for automated parking lots and conventional parking lots.

Impact of varying starting position The vehicle starting position (i.e., initial row to park when the stack is empty) has a great impact on the two automated parking strategies, specially for the SLS planning strategy. The starting position 0 corresponds to the top position in the stack, while the starting position 9 corresponds to the bottom of the stack (i.e., contiguous to the buffer zone). As expected, parking vehicles closer to the buffer area increases the number of per-vehicle maneuvers as the probability of moving these vehicle due to the entry or exit of other vehicles increases considerably. On the other hand, parking close to the buffer area allows decreasing the average and the maximum per-vehicle travel distance for SLS. The average per-vehicle travel distance for the strategy COA remains fairly constant for different starting positions. Thus, there exists a trade-off between the number of maneuvers and the total travel distance. The selection of the most appropriate planning strategy and its configuration will depend on the car park operator's objectives Note also that the performance of the two strategies is very similar for higher starting positions as the *virtual* size of the parking lot decreases. Also, the vehicle departure times increases as vehicles are further way from the buffer area.

Discussion Since our main aim was to understand the performance limits of the system, we assumed that in *COA* vehicle exit times are available and known precisely. With regard to the availability of exit times, we propose the implementation of a *prediction system* that uses the regularity of daily routines of end users [e.g.,(Nunes, Moreira-Matias, and Ferreira 2014)] or to request that information from the end user. The variability of exit times can impact the system performance due to the need of more frequent system reconfigurations to allow the early or late exit of a vehicle; these systems reconfigurations could potentially lead to a higher number of vehicle maneuvers and longer travel distances. If the user provided the exit time, this could also be held accountable for the early or late vehicle exit by paying additional parking charges or by increasing the waiting time on departure.

Conclusions

We presented planning strategies for automated, high-density parking lots. The systems allows doubling the space efficiency, while simultaneously providing fast access times and ensuring low collaborative in-park mobility. The system is specially targeted to electric vehicles due to its high energy efficiency at low speeds and despite repeated start-stop.

As future work, we will study how different parking layouts and vehicle relocation patterns impact the system performance. We also plan to generalize the herein presented techniques to other domains, while maintaining low computational complexity. We also aim to quantify how randomness on vehicle exit times impacts the system performance.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions that helped improve the quality of the manuscript. The authors also thank Zafeiris Kokkinogenis for the valuable discussions. This work was supported by the projects Hyrax (CMUP-ERI/FIA/0048/2013), S2MovingCity (CMUP-ERI/TIC/0010/2014) and UID/EEA/50008/2013.

References

Coffman, Jr., E. G.; Garey, M. R.; and Johnson, D. S. 1997. Approximation algorithms for np-hard problems. Boston, MA, USA: PWS Publishing Co. chapter Approximation Algorithms for Bin Packing: A Survey, 46–93.

Conner, D. C.; Kress-Gazit, H.; Choset, H.; Rizzi, A. A.; and Pappas, G. J. 2007. Valet parking without a valet. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 572–577.

de Wilde, B.; ter Mors, A. W.; and Witteveen, C. 2013. Push and rotate: Cooperative multi-agent path planning. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, 87–94.

d'Orey, P. M. 2017. Automated High-density Parking Lots. http:/www.dcc.fc.up.pt/ pedro.dorey/parking.html. Accessed: 2017-03-14.

Edwards, D. 2012. Cars kill cities. *Progressive Transit Blog*.

Ferreira, M.; Damas, L.; Conceico, H.; d'Orey, P. M.; Fernandes, R.; Steenkiste, P.; and Gomes, P. 2014a. Self-automated parking lots for autonomous vehicles based on vehicular ad hoc networking. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 472–479.

Ferreira, M.; Damas, L.; Conceio, H.; M. d'Orey, P.; Fernandes, R.; Steenkiste, P.; and Gomes, P. 2014b. Device and method for self-automated parking lot for autonomous vehicles based on vehicular networking. WO2015114592 A1 Patent App. PCT/IB2015/050,736 (EP3100253A1, CA2938378A1).

Hill, J.; Rhodes, G.; Vollar, S.; and Whapples, C. 2005. *Car park designers' handbook*. Thomas Telford.

Huetter, C. 2016. More Shuttles, Less Cost: Energy Efficient Planning for Scalable High-Density Warehouse Environments. In *International Conference on Automated Planning and Scheduling*.

Ma, H.; Koenig, S.; Ayanian, N.; Cohen, L.; Hoenig, W.; Kumar, T. S.; Uras, T.; Xu, H.; Tovey, C.; and Sharon, G. 2016a. Overview: Generalizations of multi-agent path finding to real-world scenarios. In *the 25th International Joint Conference on Artificial Intelligence (IJCAI) Workshop on Multi-Agent Path Finding*.

Ma, H.; Tovey, C.; Sharon, G.; Kumar, T. S.; and Koenig, S. 2016b. Multi-agent path finding with payload transfers and the package-exchange robot-routing problem. In *AAAI Conference on Artificial Intelligence*.

Nunes, R.; Moreira-Matias, L.; and Ferreira, M. 2014. Using exit time predictions to optimize self automated parking lots. In *IEEE Conference on Intelligent Transportation Systems (ITSC)*, 302–307.

Salido, M. A.; Sapena, O.; Rodriguez, M.; and Barber, F. 2009. A Planning Tool for Minimizing Reshuffles in Container Terminals. In *2009 21st IEEE International Conference on Tools with Artificial Intelligence*, 567–571.

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

Uhlemann, E. 2015. Active Safety Vehicles Evolving Toward Automated Driving [Connected Vehicles]. *IEEE Vehicular Technology Magazine* 10(4):20–23.

Wang, K.-H. C., and Botea, A. 2009. Tractable multi-agent path planning on grid maps. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, 1870–1875. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Wang, K.-H. C., and Botea, A. 2011. Mapp: A scalable multi-agent path planning algorithm with tractability and completeness guarantees. *J. Artif. Int. Res.* 42(1):55–90.

Wang, W.; Song, Y.; Zhang, J.; and Deng, H. 2014. Automatic parking of vehicles: A review of literatures. *International Journal of Automotive Technology* 15(6):967–978.

Wang, K.-H. C.; Botea, A.; et al. 2008. Fast and memory-efficient multi-agent pathfinding. In *International Conference on Automated Planning and Scheduling*, 380–387.

Yu, J., and LaValle, S. M. 2013. *Multi-agent Path Planning and Network Flow*. Berlin, Heidelberg: Springer Berlin Heidelberg. 157–173.