ParkAssistant: An Algorithm for Guiding a Car to a Parking Spot

Nemanja Djuric
Department of Computer and Information Sciences
Temple University
1805 N. Broad Street, 319 Wachman Hall
Philadelphia, PA 19122
E-mail: nemanja@temple.edu
Phone: 267-334-4708
Fax: 215-204-5082

Mihajlo Grbovic
Department of Computer and Information Sciences
Temple University
1805 N. Broad Street, 323 Wachman Hall
Philadelphia, PA 19122
E-mail: mihajlo@temple.edu
Phone: 267-252-4617
Fax: 215-204-5082

Slobodan Vucetic
Department of Computer and Information Sciences
Temple University
1805 N. Broad Street, 304 Wachman Hall
Philadelphia, PA 19122
E-mail: vucetic@temple.edu
Phone: 267-334-4708
Fax: 215-204-5082

Submitted for presentation and publication to the
95th Annual Meeting of the Transportation Research Board
January 10th-14th, 2016

Submitted on: August 1st, 2015
Resubmitted on: November 15th, 2015

Word count: 7,249 = 5,749 words + 250 · (5 pictures + 1 table)

ABSTRACT

Parking search is a major issue in urban areas. Drivers in major cities face a daily struggle in finding parking space, and much of this is due to lack of information about parking rules, parking prices, traffic conditions, and parking availability. As a consequence, drivers often perform inefficient search for a parking space and spend too much time searching, pay too much, or park too far from an intended destination. Inefficient parking search is not a problem only for drivers, it is also increasing traffic congestion and pollution and it causes a distortion of the parking market. Despite the vast technological advances in recent decades, parking search remains fundamentally the same societal problem it has been for almost a century. The objective of this paper is to address this issue by proposing the ParkAssistant, an algorithm that calculates a cruising route that minimizes the expected cost of parking, defined as a mix of price and time to reach the destination. To calculate a good cruising route, the algorithm uses parking information that consists of parking rules, traffic conditions, probabilities of finding an empty parking space, and drivers' utility function. We evaluated ParkAssistant through simulations using real-life parking occupancy data from San Francisco, CA. The results indicate that, as compared to an uninformed driver model, it allows drivers to find parking much faster. The results also show that the quality of ParkAssistant recommendations grows with the quality of parking information the method is provided with.

INTRODUCTION

Parking search is a major issue in urban areas. A recent study (*1*) conducted in 20 international cities shows that drivers face a daily struggle in finding a parking spot: 60% of drivers have abandoned their search for a spot at least once, and more than a quarter have gotten into an argument with a fellow motorist over a parking space. Much of this is caused by lack of information about parking rules, parking prices, and parking availability near the destination. Thus, drivers are often forced to perform an inefficient search for a parking spot, thus spending too much time searching, paying too much, or parking too far. Inefficient parking search is not a problem only for drivers seeking parking. The same study estimates that 30% of overall traffic in major cities originates from drivers searching for a parking spot. Thus, inefficient parking search is increasing traffic congestion and pollution. From the economic perspective, inefficient parking search also causes a distortion of the parking market. Interestingly, despite the vast technological advances during the recent decades, parking search remains fundamentally the same societal problem it has been for almost a century (*2*). The objective of this work is to become a stepping stone towards solving the parking search problem.

Let us look at the setup of urban parking in more detail. There are two major types of parking: on-street and off-street. The on-street parking spots are typically alongside the roads and are treated as public property. The off-street parking includes parking lots and garages that are managed either by private providers or by municipal authorities. Both types of spots are regulated with parking rules defining availability, time limits, pricing, and permissions, and there is typically a large variability in rules even between spatially neighboring locations. Regarding the pricing, it is typical in the U.S. cities that the on-street parking is significantly more affordable than the off-street parking, particularly for residential parking and for shorter stays of up to a couple of hours. In fact, a vast majority of on-street spots are typically free and lightly regulated, and only the spots within business, tourist, and entertainment areas are metered and could have complex regulations. Affordability of on-street spots is a consequence of a political view that treats them as a shared public good. Prices in parking garages and lots are typically market-based, reflecting the fact that construction and maintenance of parking spaces is costly. As a consequence, the on-street spots are in higher demand and drivers are typically interested in exploring their availability before considering the off-street parking. Due to the higher demand and occupancy rates of the on-street parking, searching for an on-street parking spot often requires significantly more time and is more uncertain. This opens an interesting dilemma – whether saving several dollars is worth spending an uncertain amount of time searching for an on-street spot. This dilemma is at the core of the decision challenges that many drivers face when considering parking in urban areas.

To better understand the complexity of parking decision-making from the perspective of drivers, let us look at all the relevant information that one should consider. Let us assume a car is approaching its destination and its driver needs to decide where to search for a parking spot. The first piece of information is *where the parking is permitted* in the neighborhood of the destination and *what the prices are*. The second piece of information is current *availability of parking* at the permitted locations. In particular, it is appropriate to think about the availability in terms of probability of finding an open parking spot at the time of the arrival. The third piece of information that impacts the time needed to find parking are *traffic conditions*; in particular, the speed of traffic. The fourth piece of information is related to the *driver's parking preferences*. This is related to questions such as the purpose of the trip, the time constraints, driver's willingness to walk, their budget, and the comfort with on-street parking (e.g., parallel parking may be an issue for some drivers). Importantly, even if all that diverse and complex information were available, it is not clear how to present it to the driver to facilitate the parking search. Due to the uncertain nature of parking search, processing all the available information and reasoning about it to make parking decisions could be a daunting task for any driver. In practice, the parking information is likely to be incomplete and uncertain, which further exacerbates the complexity of the parking decision-making process.

BACKGROUND

Recent years have seen the birth of "smart parking", which refers to the innovative use of technology to make parking more efficient and easier. In the following, we briefly review efforts that are in our view the most relevant to the current work and which motivate our approach. One type of effort focuses on parking occupancy monitoring. For example, many parking garages have sensing systems that provide real-time information about parking availability, such as a total number of available parking spots in the garage or a number of available spots in each garage segment. For on-street parking, several cities have been experimenting with sensors installed at the individual on-street parking spots to collect real-time information about parking availability. For example, SFpark (*3*) is a federally funded pilot project initiated in 2011, which resulted in installation of occupancy sensors at around 7,000 metered parking spots in downtown San Francisco, CA. Their purpose is to help determine demand-responsive pricing and to provide real-time occupancy and pricing information to the public through Internet. While parking sensors indeed give very useful information, this comes at a price. For example, StreetLine that installed similar sensors at around 10,000 similar on-street parking sensors in Los Angeles apparently charges around $1 per day for operation and maintenance of each sensor (*4*). A recent study by the San Francisco Department of Transportation found that there are 26,750 parking meters, 275,450 total on-street spots, and another 166,500 off-street parking spots in the city, and that the length of all street parking spots exceeds the total length of California's entire 840-mile coastline (*5*). Having this in mind, it is doubtful that placing dedicated sensors even under a fraction of the spots could be cost-effective.

There are potential alternatives to sensors embedded at parking spots. A notable example is ParkNet (*6*), which observes on-street parking occupancy from a moving vehicle equipped with a GPS receiver and a passenger-side-facing ultrasonic rangefinder. The authors claim that their system is 95% accurate and that it might be cheaper than the SFpark sensing project if the sensors were installed on all taxis in San Francisco. However, the deployment of such a system would still require significant investment by the municipalities and the resulting coverage would be uneven and dependent on the taxi routes, which might not correlate well with the locations of the most demanded parking spots. Other authors studied a scenario where multiple drivers use an app that collects their GPS traces, detects when somebody parked or left a parking spot, and broadcast this information back to the users (*7*, *8*, *9*). In (*10*) the authors propose how to use such information to estimate parking probabilities by assuming the parking data are obtained by random sampling.

Beyond occupancy sensing, creation of traveler information systems for parking is another important smart parking challenge. These systems typically come in the shape of websites or apps that provide real-time parking information. For example, a popular ParkMe app (*11*) provides pricing

regulations and rules for all parking lots and garages in a number of cities. It also provides occupancy information for garages equipped with smart sensors, as well as navigation to the selected garage. However, in most cities, the app does not provide any information about the on-street parking, thus making it an incomplete source of information. For cities such as San Francisco and Los Angeles, during 2012 and 2013 it was possible for public to view real-time occupancy of monitored spots, but the coverage was limited to a small number of city blocks. Other than the occupancy, parking rules are another important source of information. Several cities such as Seattle and New York provide web interfaces that allow querying of the on-street rules. Interestingly, parking signs in New York City are so complicated that drivers have trouble interpreting them, and the city maintains several dedicated web pages helping the public to interpret them (*12*). There remain two major open questions related to the traveler information systems for parking: how to collect information necessary for informed parking, and how to present this information to drivers to help them minimize the cost of their parking search.

A question of parking reservation and pricing has also garnered significant research interest. Some researchers are proposing systems that allow drivers to reserve open parking spots (*13*, *14*, *15*). In addition, some authors take a game-theoretic view of parking search and propose real-time auctions to determine who gets an open spot (*15*, *16*, *17*), which often requires decisions to be centralized and information to be censored (*16*). However, such reservation and pricing approaches use questionable assumptions that open spots can be detected, that they could be reserved, and that all drivers have access to the auction system. A related approach is to set parking prices in a demand-responsive way such that at least a small fraction of spots are available on every street at any given time (*18*). This approach was tested in San Francisco as part of the SFpark project, and it was observed that the resulting average parking prices were lower, parking availability was improved, and drivers were spending less time looking for parking (*19*). Despite this apparent success, some scholars are questioning the stated benefits and point to the remaining open issues (*20*, *21*), and it is unclear if the approach can be translated to other cities because it implies significant investments in parking sensor networks, as well as a political will to implement the unpopular demand-responsive pricing.

The question of parking choice and parking search has been studied before in the context of agent-based modeling of parking. In (*22*), a parking choice model was proposed, which models how drivers decide which parking spot to go to. Similarly to our work, the authors account for driver's budget, walking distance, and other relevant factors, and assign a utility score to each parking location. On the other hand, the authors model the decision process while not addressing the problem of a parking search, which is the topic of this study. In ParkAgent (*23*), a simplistic model of parking search is proposed where drivers drive directly to the destination, start searching as they get close to it, and continue going in circles until they find a parking spot. This model is similar to our uninformed driver model that we will use as a baseline in our experiments. In (*24*), the ParkAgent model has been enhanced to use sensor information about parking occupancy. As compared to our optimization-based model, the model from (*24*) is a greedy search heuristics that might not fully exploit the available occupancy information.

PROPOSED APPROACH

As we described in the previous sections, despite many efforts it is still not clear how to harness novel, advanced technology and create smart parking solutions that address the longstanding parking challenges. We believe that an answer to most of the open questions in smart parking has to start with helping drivers find the best parking spot they could, done by using all the publicly available information. If we can help every driver maximize parking utility based on their personal preferences, we believe it can start a cascade of events that will minimize driver frustration, reduce parking search time, create an efficient parking market, allow policy makers to make informed decisions about parking, and enable benefiting from future technological advances. This is why the aim of this work was to propose an algorithm that uses uncertain and incomplete parking information to help drivers

find a parking spot that maximizes their expected utility. We will refer to this algorithm as the *ParkAssistant*, and describe it in the following section.

Methodology

Parking search is a complex decision problem under uncertainty, where it is possible that several approaches are viable. To compute the optimal one, ParkAssistant first acquires parking instructions from a driver consisting of current car location (origin), location of destination, and parking preferences such as intended parking duration, time flexibility, price elasticity, and willingness to walk. The parking preferences are converted into an appropriate utility function. ParkAssistant then uses the obtained parking instructions to recommend a parking route defined as a sequence of M consecutive road segments starting from the current location, where each segment is labeled either as PARK or NO PARK. Value M is fixed and prespecified by a modeler, usually set to an appropriate number to make sure that, for example, the route has a length of at least 10 city blocks. The driver is instructed to drive along the parking route and to park at the first available parking spot along road segments labeled with PARK, while ignoring potentially available parking spots at NO PARK segments (e.g., the segment may be far from the destination and driver's willingness to walk is low).

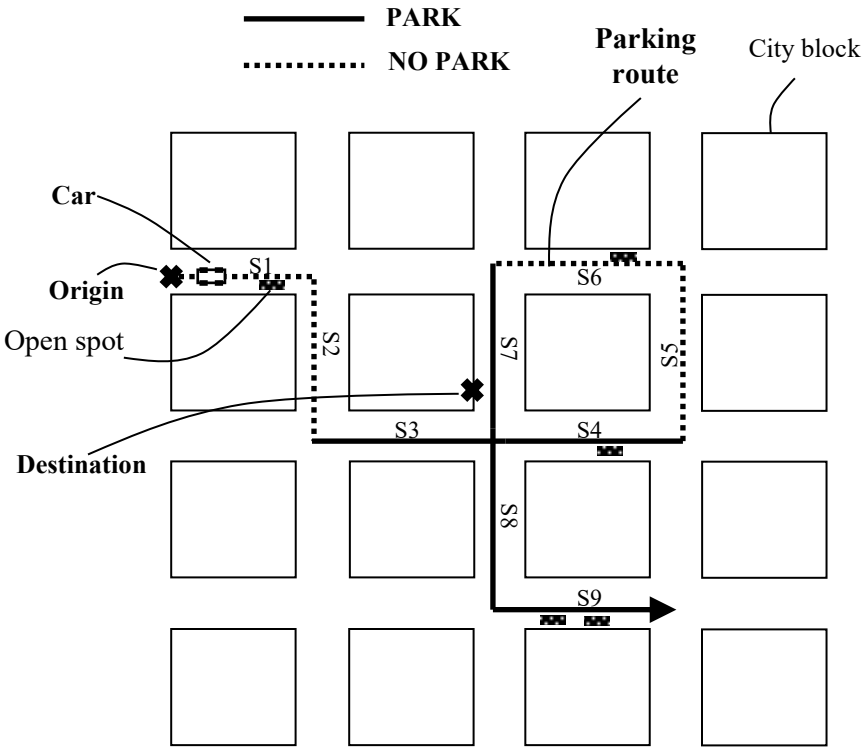

FIGURE 1 Illustration of the parking search process using ParkAssistant

Figure 1 provides an illustration of the parking search process. It shows a city grid with 4 times 4 city blocks. There are only 6 open parking spots plotted as small checkered rectangles. The car is at the origin and its driver is interested in reaching the destination. The driver does not know in advance where the open spots are. ParkAssistant provided a parking route of length $M = 9$, which consists of full and dashed lines denoting PARK and NO PARK street segments, respectively. The car starts moving along segment S1. Since that part of the route is dashed, it means the driver ignores any open parking spots and keeps driving. Thus, the car does not park at the open spot along segment S1. Since segment S2 is also dashed, the driver keeps following the route without looking for parking.

Upon reaching segment S3, which is a full line, the driver keeps looking for the open parking spots, but there are not any. Upon reaching the full-line segment S4, the driver keeps looking and observes the open parking spot. The driver parks the car at the open spot, potentially pays the parking fee, and proceeds walking towards the destination. Given the outcome, the question is what the driver's satisfaction with the provided route is. We propose to measure it using a utility function.

Let us denote the utility function that quantifies drivers' satisfaction with parking and reaching the destination from the origin in t minutes as $U(t)$. In Figure 2 we illustrate two possible utility functions which have the maximum value of 1 when $t = 0$, and equal to 0 for $t > 20$ min. The first function is constant until $t = 20$ min, while the second function gradually decreases towards zero. The first utility function may correspond to a driver who must reach the destination in 20 minutes because of an important business meeting, where the utility drops significantly as the meeting time approaches, thus favoring parking as soon and as close as possible. The second utility function may correspond to a driver interested in going to a theater with friends, where arriving any time before the theater door closes is acceptable, thus allowing less-constrained search and more room for exploration. Given the function $U(t)$, it is possible to calculate the utility of parking at any particular parking spot. For example, the utility may be calculated as $U(T_d + T_w)$, where T_d is driving time from the origin to the parking spot and T_w is walking time required to reach the destination from the parking spot. For the example route given in Figure 1, the utility equals $U(T_d(\{S1, S2, S3, S4\}) + T_w(S4, \text{destination}))$, where $T_d(S)$ is a total driving time along segments from the segment multiset S, and $T_w(A, B)$ is a total walking time from location A to location B.

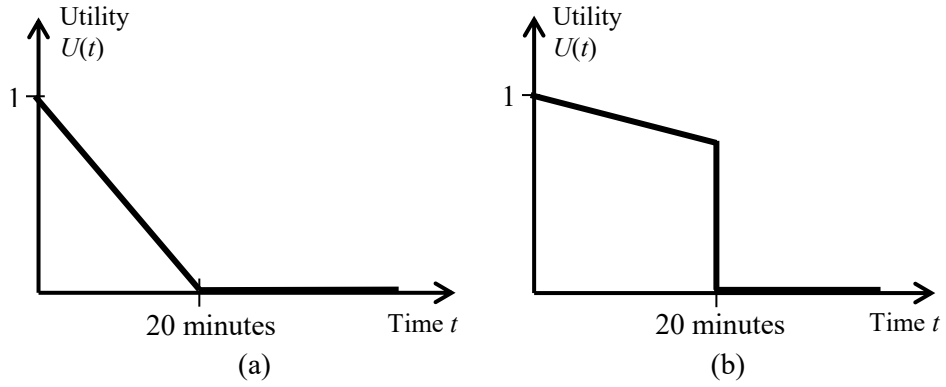

FIGURE 2 Examples of utility functions $U(t)$ that simulate: (a) driver going to an important business meeting; (b) driver going to a theater with friends

The proposed approach is very flexible, and a modeler can use more elaborate ways to calculate the utility in order to better model drivers' parking preferences. For example, the utility may be calculated as $U(T_d + w_1 \cdot T_w)$ where w_1 is a constant larger than one, reflecting that the driver prefers to find a parking spot close to the destination even if it requires longer total driving time. It is also possible to account for the price elasticity of the driver. For example, the utility may be calculated as $U(T_d + w_1 \cdot T_w + w_2 \cdot price)$, where *price* is the price of parking and w_2 is a positive constant, thus reflecting that the utility drops with the price of parking. Alternatively, the utility can be defined as a function of parking search time and price, which may also depend on time of day and weather conditions.

Calculating parking route utility

Given the ability to calculate the utility for parking at a specific spot, it becomes possible to calculate the utility of a parking search route. If the locations of open parking spots were known in advance, calculating the utility of a route would amount simply to observing the first open spot along

the PARK segments of the route, calculating T_d and T_w, and computing the utility as $U(T_d + T_w)$. However, unless parking spots are monitored with sensors (which is very costly), the parking occupancy could not be known with certainty. In addition, knowledge about the traffic conditions is also typically uncertain, which makes it challenging to calculate driving time T_d. Since the utility is subject to uncertainties such as actual traffic and the probabilities of finding available parking spots along the route, ParkAssistant calculates the expected utility for the given parking route. Let us represent the unique identifiers (IDs) of road segments along a specific route of length M as a sequence $\{i_1, i_2, \ldots, i_M\}$. Then, let us define as $\{park_1, park_2, \ldots, park_M\}$ the corresponding indicators of whether the driver should look for a parking spot along each of the route segments, where $park_m = 1$ encodes that the m^{th} segment along the route with ID i_m is labeled as PARK and $park_m = 0$ means that the segment is labeled as NO PARK. Let us denote with $P(i_M)$ the probability that there is an open parking spot along street segment i_M.

To get an intuition on how to calculate the expected utility of parking route of length M defined with a sequence of segments $\{i_1, i_2, \ldots, i_M\}$ labeled with $\{park_1, park_2, \ldots, park_M\}$, let us assume that $park_1 = 1$. Then, with probability $P(i_1)$, the car will park along the first segment and the utility will equal $U(T_d(i_1) + T_w(i_1, dest))$, where $dest$ denotes the destination. If we assume $park_1 = 0$, then the probability of parking along the first segment is 0. Thus, we can claim that with probability $P(i_1) \cdot park_1$ the utility will equal $U(T_d(i_1) + T_w(i_1, dest))$, and with probability $1 - P(i_1) \cdot park_1$ the car will continue driving. Following this reasoning, we can conclude that with probability $(1 - P(i_1) \cdot park_1) \cdot P(i_2) \cdot park_2$ the car will park along segment i_2 and observe utility $U(T_d(\{i_1, i_2\}) + T_w(i_2, dest))$. By continuing this line of thought, it becomes possible to express the lower bound U_{lower} on the expected utility of the route as the following recursive formula,

$$\begin{aligned} U_{lower}(\{i_1,i_2,...,i_M\},\{park_1,park_2,...,park_M\}) = \\ U_{lower}(\{i_1,i_2,...,i_{M-1}\},\{park_1,park_2,...,park_{M-1}\}) + \\ P(i_M)\cdot park_M \cdot U\left(T_d(\{i_1,i_2,...,i_M\}) + T_w(i_M,dest)\right)\cdot \prod_{m=1}^{M-1}\left(1-P(i_m)\cdot park_m\right). \end{aligned} \tag{1}$$

The formula expresses a lower bound U_{lower} of the expected utility, because it assumes that if parking is not found along the M segments of the route, the utility of the subsequent parking search is zero. Thus, the lower bound represents a pessimistic estimate of the expected utility. To get an optimistic estimate, we can calculate the upper bound U_{upper} as follows,

$$\begin{aligned} U_{upper}(\{i_1,i_2,...,i_M\},\{park_1,park_2,...,park_M\}) = \\ U_{lower}(\{i_1,i_2,...,i_M\},\{park_1,park_2,...,park_M\}) + \\ U\left(T_d(\{i_1,i_2,...,i_M\}) + T_d(i_M,dest)\right)\cdot \prod_{m=1}^{M}\left(1-P(i_m)\cdot park_m\right), \end{aligned} \tag{2}$$

where $T_d(i_M, dest)$ denotes the driving time from the M^{th} route segment to the destination. The upper bound in the equation assumes that if there are no available parking spots along the M segments of the route, the driver will be able to drive directly to the destination and find a parking spot there. Both bounds approach the expected utility when M is large.

Computing the optimal parking route

The objective of ParkAssistant is to find the route of length M with the highest expected utility. If we are interested in finding the optimal route in a brute force manner, this would require examination of all routes of length M, whose number scales exponentially as $\mathcal{O}(2^M)$. Instead, we propose a computationally efficient, greedy procedure for finding the optimal route, given as Algorithm 1.

The recursive algorithm is invoked twice for each parking instance. In particular, the algorithm is invoked with the segment on which the origin is located as an input, labeled once with PARK and once with NO PARK label. In addition, all global variables are initialized to the values given in the table before the function is run. The function first checks if the length of the current route has reached the prespecified length M (line 01), in which case it updates the current best route information if the route's lower bound on the expected utility is higher than the lower bound of the best route so far (lines 03-06), and the algorithm then brakes out from the recursion (line 07). If the route length is still smaller than M, it is expanded with the segments that can be reached from the final segment of the route (line 09), and the method is called recursively for each of the expanded routes and for both possible park labels (lines 10-17). However, before the recursive call, for each expanded route we first check if the upper bound is lower than the lower bound of the best route so far (line 15), which allows us to prune away bad routes. In practice, we found that the pruning step reduces the search space significantly and results in large computational speed ups, making the route optimization practical for M as large as 20. Note that if the utility function is monotonically non-increasing, such as utility functions given in Figure 2, it can be shown that the computed route of length M is guaranteed to be optimal, and to have the highest lower bound on the expected utility of all the possible routes of length M.

ALGORITHM 1 Pseudocode of the recursive routing algorithm

```
ParkAssistant(route = segment of origin, labels = 0/1)
Input variables : current route route, park/no park labels labels;
Global variables: optimal route bestRoute = [], optimal labels
                  bestLabels = [], lower bound of current optimal
                  route maxLowerBound = 0, max route depth M;

00: // break out of recursion if maximum depth is reached
01: if (route is of length M)
02:    // check if this is a new best route
03:    if (U_lower of route and labels > maxLowerBound)
04:       maxLowerBound ← U_lower;
05:       bestRoute     ← route;
06:       bestLabels    ← labels;
07:    return;

08: // expand the current route
09: find set S of next driving segments for route;
10: foreach (segment seg from S)
11:    foreach (label lab from {0, 1} set)
12:       newRoute  ← append seg to route;
13:       newLabels ← append lab to labels;
14:       // check if we can prune this route
15:       if (U_upper of newRoute and newLabels < maxLowerBound)
16:          next;
17:       ParkAssistant(newRoute, newLabels);
```

Let us summarize the inputs ParkAssistant relies on to provide the route. First, ParkAssistant needs to know the origin and the destination. Second, it needs to know parking preferences of the driver to set the utility function. Third, it needs to know the road network, traffic rules (e.g., one-way streets, no-turns), and parking regulations. Fourth, it needs to know the current traffic conditions to calculate the driving time. Finally, it needs to know parking probabilities for all segments within the area of interest. We implemented ParkAssistant and evaluated it in San Francisco, CA. In the following section we present the experimental results that illustrate the performance of the algorithm.

EXPERIMENTS

In this section we present results of an empirical evaluation of ParkAssistant. Our evaluation was focused on comparison with a naïve, uninformed human agent, as well as on examining the impact of incomplete information on the performance of ParkAssistant. To evaluate the algorithms, we used a public SFPark API to collect a data set containing minute-by-minute parking occupancy information obtained from SFpark sensors at 247 blocks in downtown San Francisco, location of which is shown in Figure 3. For this study we considered occupancy data from 4 consecutive Thursdays in August 2013. The first three Thursdays were used as training data, while the last Thursday was used for testing. For segments not covered by the sensors we assumed that parking is not available. Although this assumption does not hold in practice, it nevertheless allowed us to properly evaluate ParkAssistant in a controlled setting.

FIGURE 3 Locations of parking sensors in downtown San Francisco, CA (shown as red dots)

To test the algorithms, we generated 10,000 parking instances. For each instance we randomly selected an origin location in downtown San Francisco and a destination location such that it is within 0.15 miles of the origin (0.15 miles is roughly equivalent to 5 city blocks). For each parking instance we uniformly at random selected a starting time between 4pm and 5pm, when parking spots are in high demand in the downtown. We then calculated parking routes for each parking instance using four algorithms:

(a) "*Uninformed driver*" that simulates uninformed driver who drives around the desired destination (driving straight for a few street segments, and then flipping a fair coin and turning double left or double right to continue driving parallel to the starting direction; after the double turn when the route passes closest to the destination another fair coin is flipped and the route continues straight, left, or right, after which the process is repeated from the beginning);

(b) "*All 0.9*" that uses ParkAssistant where 90% parking probability is assumed on all street segments, including both those that are sensor-equipped and those that are not;

(c) "*Park 0.9*" that uses ParkAssistant where 90% parking probability is assumed on street segments equipped with the sensors, and 0 elsewhere (this corresponds to our assumption that parking is available only on sensor-equipped segments);

(d) "*Historical*" that uses ParkAssistant where parking probabilities are estimated from the training data (the parking occupancy between 4pm and 5pm from the first three Thursdays).

Given a suggested route, we let the car virtually drive along the route and park on the first sensor-equipped segment which is not fully occupied. To know if a street segment is full, we check its parking occupancy at a time of the car arrival. Then, for each parking instance we calculate the time

to reach the destination, which is a sum of the driving and walking times. In the experiments we assumed the constant cruising speed of 10mph and the constant walking speed of 2mph. In addition, we assumed the parking is free and that the utility function is the one depicted in Figure 2(a).

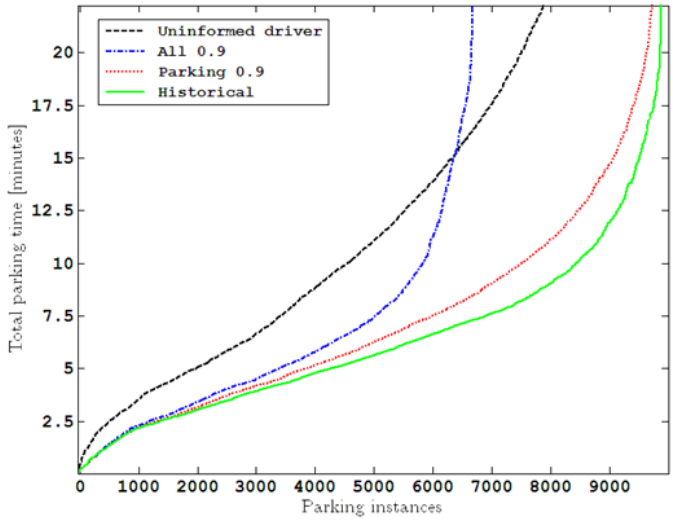

FIGURE 4 Comparison of various approaches to compute parking route

Comparison of the four algorithms is shown in Figure 4, where we ordered the parking instances by ascending parking times. We found that the average parking times were 17.08 minutes for "Uninformed driver", 12.46 minutes for "All 0.9", 7.31 minutes for "Park 0.9", and 6.24 minutes for "Historical" approach. Moreover, the figure shows that "Uninformed driver" finds a parking spot in less than 5 minutes in only 18% of parking instances, while "Historical" manages to do this in 42% of the instances, constituting a significant increase in parking efficiency over the baseline. It is also interesting to note that the average parking time difference between "Park 0.9" and "Historical" is only around one minute, which is somewhat surprising considering that "Park 0.9" did not consider any historical training data. In addition, we observe that the "Historical" method is significantly more difficult and expensive to implement in practice, requiring installation of parking sensors. This result suggests that inexpensive "Park 0.9" represents a viable and effective alternative to potentially expensive "Historical" approach, and can be used in cases when historical data are not available.

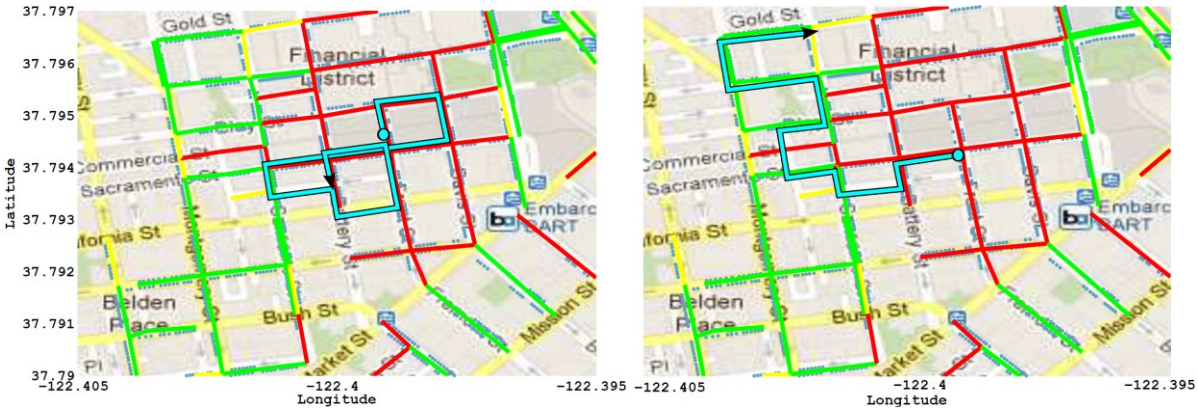

FIGURE 5 Parking routes (shown in blue) found by: a) "All 0.9"; b) "Historical"; circle represents an origin which is also a destination, red, yellow, and green denote historical availabilities on segments (<33%, 33-66%, and >66% parking spots available, respectively)

To better illustrate differences between various approaches, in Figure 5 we show routes computed by "All 0.9" and "Historical" methods for the same parking instance initiated at 4:55pm. The circle denotes the starting location which is also a desired destination, blue line denotes the suggested route, while red, yellow, and green colors denote historical parking availabilities on parking segments for this time (representing less than 33%, between 33% and 66%, and more than 66% parking spots available, respectively). We can observe a significant difference between the two routes. Since "All 0.9" was not aware of the historical parking probabilities, the method suggested the route that led a user to circle around the destination and search for parking at street segments where the parking probabilities were very low and made it unlikely to find a spot. On the other hand, "Historical" suggested the route that covers many segments with relatively higher probability of finding a parking spot, while still keeping the user as close as possible to the destination in order to minimize the walking time.

The results clearly indicate that ParkAssistant can substantially reduce parking time and take advantage of any parking information available. The computational cost of the proposed route-finding algorithm is small, allowing it to be implemented and run directly on smartphone devices.

CONCLUSION

Finding a parking spot in urban jungles has been an issue for millions of drivers in the past decades, and is becoming an even larger societal and environmental concern with further increase of populations in metropolitan regions. In this paper we proposed an approach that addresses this growing problem, designed to help the drivers find the most effective and efficient parking route in urban areas. The method, called ParkAssistant, is capable of taking into consideration current parking and traffic rules and conditions, as well as driver's preferences such as the specific purpose of the trip and willingness to walk, and to compute an optimal parking route that maximizes their overall parking utility. To this end we defined the parking utility function, and described how to efficiently find the route that maximizes the expected utility. The proposed method was evaluated on the real-world data collected by the SFpark project from San Francisco, CA, where we compared ParkAssistant to a naïve baseline and explored how the performance is affected by varying levels of available parking information provided to the algorithm. The results strongly suggest the benefits of the proposed approach, and present a step towards solving the problem of parking in urban areas.

REFERENCES

1. IBM. *IBM Global Parking Survey: Drivers Share Worldwide Parking Woes*. https://www-03.ibm.com/press/us/en/pressrelease/35515.wss. Accessed November 13, 2015.
2. Shoup, D. C., Cruising for parking. *Transport Policy*, vol. 13, no. 6, pp. 479-486, 2006.
3. SFPark. http://www.sfpark.org. Accessed November 13, 2015.
4. Eilene Zimmerman. CNNMoney. *A silver bullet for urban traffic problems*. http://money.cnn.com/2011/04/29/technology/streetline/. Accessed November 13, 2015.
5. Kurtis Alexander. SFGATE. *How many parking spots are there in S.F?* http://blog.sfgate.com/stew/2014/05/23/how-many-parking-spots-are-there-in-s-f/. Accessed November 13, 2015.
6. Mathur, S., Jin, T., Kasturirangan, N., Chandrasekaran, J., Xue, W., Gruteser, M., Trappe, W., Parknet: Drive-by sensing of road-side parking statistics, *International Conference on Mobile Systems, Applications, and Services*, pp. 123-136, San Francisco, USA, 2010.
7. Stenneth, L., Wolfson, O., Xu, B., Philip, S. Y., PhonePark: Street parking using mobile phones, *International Conference on Mobile Data Management*, pp. 278-279, 2012.
8. Koster, A., Oliveira, A., Volpato, O., Delvequio, V., Koch, F., Recognition and recommendation of parking places, *Advances in Artificial Intelligence*, pp. 675-685, 2014.

9. Nawaz, S., Efstratiou, C., Mascolo, C., ParkSense: A smartphone-based sensing system for on-street parking, *Annual International Conference on Mobile Computing and Networking*, pp. 75-86, Miami, USA, 2013.
10. Xu, B., Wolfson, O., Yang, J., Stenneth, L., Yu, P. S., Nelson, P. C., Real-time street parking availability estimation, *International Conference on Mobile Data Management*, vol. 1, pp. 16-25, 2013.
11. ParkMe. http://www.parkme.com. Accessed November 13, 2015.
12. The Official Website of the City of New York. *Parking Signs and Locator*. http://www1.nyc.gov/nyc-resources/service/2178/parking-signs-and-locator. Accessed November 13, 2015.
13. Alhammad, A., Siewe, F., Al-Bayatti, A. H., An infostation-based context-aware on-street parking system, *International Conference on Computer Systems and Industrial Informatics*, pp. 1-6, 2012.
14. Wang, H., He, W., A reservation-based smart parking system. *Computer Communications Workshops*, pp. 690-695, 2011,
15. Ayala, D., Wolfson, O., Xu, B., Dasgupta, B., Lin, J., Parking slot assignment games, *International Conference on Advances in Geographic Information Systems*, pp. 299-308, 2011.
16. Kokolaki, E., Karaliopoulos, M., Stavrakakis, I., Leveraging information in parking assistance systems, *IEEE Transactions on Vehicular Technology*, 62.9: pp. 4309-4317, 2013.
17. Ayala, D., Wolfson, O., Xu, B., DasGupta, B., Lin, J., Pricing of parking for congestion reduction, *International Conference on Advances in Geographic Information Systems*, pp. 43-51, 2012.
18. Shoup, D.C., The high cost of free parking, Planners Press, Washington, DC, 2005.
19. Pierce, G., Shoup, D., Getting the prices right: An evaluation of pricing parking by demand in San Francisco, *Journal of the American Planning Association*, vol. 79, no. 1, pp. 67-81, 2013.
20. Millard-Ball, A., Weinberger, R., Hampshire, R., Comment on Pierce and Shoup: Evaluating the impacts of performance-based parking, *Journal of the American Planning Association*, vol. 79, no. 4, pp. 330-336, 2013.
21. Pierce, G., Shoup, D., Response to Millard-Ball et al.: Parking prices and parking occupancy in San Francisco, *Journal of the American Planning Association*, vol. 79, no. 4, pp. 336-339, 2013.
22. Waraich, R., Axhausen, K., Agent-based parking choice model, *Transportation Research Record: Journal of the Transportation Research Board,* vol. 2319, pp. 39-46, 2012.
23. Benenson, I., Martens, K., Birfir, S., PARKAGENT: An agent-based model of parking in the city, *Computers, Environment and Urban Systems*, vol. 32, no. 6, pp. 431-439, 2008.
24. Tasseron, G., Martens, K., van der Heijden, R., The Potential Impact of Vehicle-to-Vehicle and Sensor-to-Vehicle Communication in Urban Parking, *Intelligent Transportation Systems Magazine*, IEEE, vol. 7, no. 2, pp. 22-33, 2015.