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

Nemanja Djuric, Mihajlo Grbovic, Slobodan Vucetic

Department of Computer and Information Sciences, Temple University

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, 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, 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.

Introduction

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 a 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.

Figure 1. ParkAssistant helps drivers with parking headache

Proposed approach

ParkAssistant

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:

- current car location (origin);
- location of destination;
- parking preferences (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 (see Figure 2). 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).

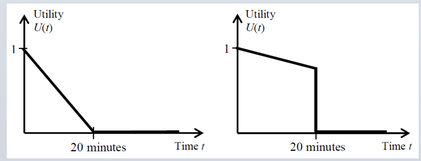

Figure 3. Examples of utility functions $U(t)$ that simulate: (a) driver going to an important business meeting; (b) driver going to a theater with friends

Finding the optimal route

In order to quantify the goodness of a route, 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 3 we illustrate two 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.

Lower bound U_{lower} on the expected utility of a route is found using the recursive formula (T_d - driving time, T_w - walking time, $i_1 \dots i_M$ - parking segments, $park_1 \dots park_M$ - parking indicators, $P(i_m)$ - parking probability),

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

To get an optimistic estimate, we calculate an upper bound U_{upper} as,

$$U_{upper}(\{i_1,i_2,\dots,i_M\},\{park_1,park_2,\dots,park_M\}) = U_{lower}(\{i_1,i_2,\dots,i_M\},\{park_1,park_2,\dots,park_M\}) + U(T_d(\{i_1,i_2,\dots,i_M\}) + T_d(i_M,dest))\cdot\prod_{m=1}^{M}(1-P(i_m)\cdot park_m)$$

Then, we can use a greedy Algorithm 1 to find an optimal route.

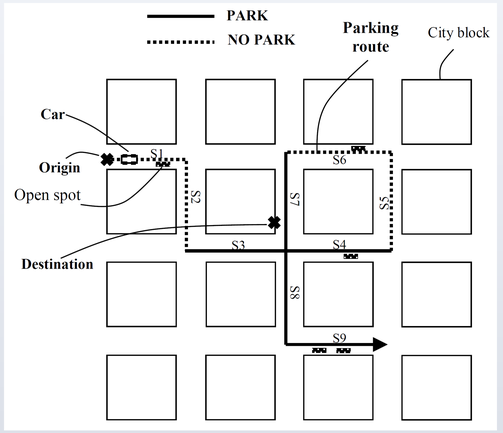

Figure 2. Example output of ParkAssistant

```
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);
```

Figure 4. Recursive algorithm for finding an optimal parking route

Data set

We used a public SFPark API to collect minute-by-minute parking occupancy information obtained from SFpark sensors at 247 blocks in downtown San Francisco. 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.

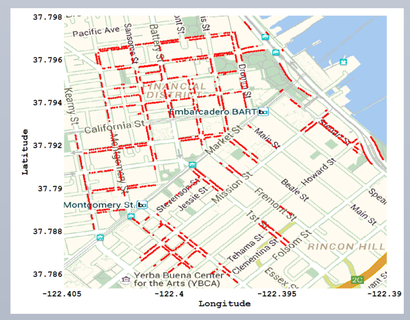

Figure 5. Locations of parking sensors in downtown SF

Experimental results

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

1. **"Uninformed driver"** that simulates uninformed driver who drives around the desired destination;
2. **"All 0.9"** that uses ParkAssistant where 90% parking probability is assumed on all street segments;
3. **"Park 0.9"** that uses ParkAssistant where 90% parking probability is assumed on segments equipped with the sensors, and 0 elsewhere;
4. **"Historical"** that uses ParkAssistant where parking probabilities are estimated from the training data.

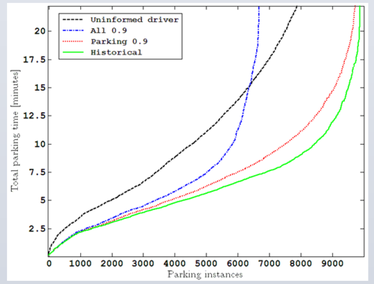

Figure 6. Comparison of various approaches to compute parking route

- 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.
- Interestingly, the average parking time difference between "Park 0.9" and "Historical" is only around one minute, although "Park 0.9" did not consider any historical training data. In addition, "Historical" method is significantly more difficult and expensive to implement in practice.

To illustrate differences between approaches, in Figure 7 we show routes by "All 0.9" and "Historical" for the same parking instance.

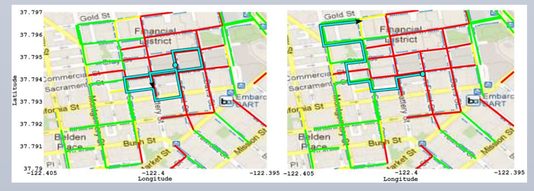

Figure 7. Parking routes (shown in blue) by: a) "All 0.9"; b) "Historical"; circle is an origin and a destination, red, yellow, and green denote historical availabilities (<33%, 33-66%, and >66% parking spots available, respectively)

- Since "All 0.9" was not aware of the historical probabilities, the method suggested the route that led a user to circle around the destination and search for parking where the parking probabilities were very low.
- "Historical" suggested a route that covers many segments with relatively higher probability of finding a parking spot, while keeping the user as close as possible to the destination to minimize walking time.