AI Communications 32 (2019) 113–124
DOI 10.3233/AIC-180574
IOS Press

Routing for on-street parking search using probabilistic data

Tobias Friedrich *, Martin S. Krejca, Ralf Rothenberger, Tobias Arndt, Danijar Hafner, Thomas Kellermeier, Simon Krogmann and Armin Razmjou
Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
E-mails: tobias.friedrich@hpi.de, martin.krejca@hpi.de, ralf.rothenberger@hpi.de

Abstract. A significant percentage of urban traffic is caused by the search for parking spots. One possible approach to improve this situation is to guide drivers along routes which are likely to have free parking spots. The task of finding such a route can be modeled as a probabilistic graph problem which is NP-complete. Thus, we propose heuristic approaches for solving this problem and evaluate them experimentally. For this, we use probabilities of finding a parking spot, which are based on publicly available empirical data from TomTom International B.V. Additionally, we propose a heuristic that relies exclusively on conventional road attributes. Our experiments show that this algorithm comes close to the baseline by a factor of 1.3 in our cost measure. Last, we complement our experiments with results from a field study, comparing the success rates of our algorithms against real human drivers.

Keywords: Parking search, probabilistic routing, constrained optimization, field study

1. Introduction

Three out of four urban citizens find the search for a parking spot the most stressful aspect in their everyday life. This situation is found to be even more stressful than the waiting time spent at public offices, as shown by two representative studies of Bitkom Research [7]. Besides the stress and time factors involved, pollution is a further crucial issue to consider in the parking place search. Up to 30 % of urban congestion can be attributed to the search for an on-street parking spot [12]. One solution to this problem might be off-street parking facilities, such as car parks and parking lots. However, these alternatives are usually expensive and their availability is limited. For this kind of parking, many solutions to find near off-street parking facilities already exist in the form of websites and several smart phone apps.[1] Since on-street parking makes up the majority of parking spots in most cities [3], it is important to investigate how to use it efficiently. Therefore, this article will concentrate on finding a curbside parking spot.

Live data from dedicated sensors monitoring parking spots have been tested [14] for applicability but do not scale. An alternative focuses on already existing historical data of urban traffic. With this data, it is possible to determine how likely it is to find a parking spot on any given road.

Lately, the interest in such parking probabilities has increased. Google proposed a machine learning approach for predicting such probabilities, based on data from Android devices.[2] Toyota has a patent with a similar approach [13]. TomTom International B.V. has a publicly available API[3] with parking probabilities and the average time spent searching for a parking place on a per-street basis, covering 105 cities worldwide, using data from 550 million of their devices. We complement this trend by proposing a solution that uses these parking probabilities and suggests roads to the user with a high probability of available parking spots.

Using the parking data from TomTom for the area of Berlin, Germany, we consider a routing problem, modeled as a probabilistic graph. Our goal is to find a route where there is a high probability of finding a parking spot. Further, the desired route should be

*Corresponding author. E-mail: tobias.friedrich@hpi.de.
[1]For example, parkopedia.com.
[2]https://ai.googleblog.com/2017/02/using-machine-learning-to-predict.html
[3]https://developer.tomtom.com/on-street-parking

optimized with respect to a short driving duration as well as a short walking distance to the desired destination, resulting in a bi-objective optimization problem. We describe algorithmic approaches for calculating optimized parking search routes, and we analyze those approaches within different experiments. Last, we present results of a practical field study that verify the knowledge gained in the experiments.

This article extends our prior work [2], where we tested our algorithms only in computer simulations. In this article, we complement these results with our field study.

2. Related work

Routing on conventional road networks has been subject to extensive research. Especially the approach of route queries on probabilistic graphs has recently gained increasing attention. More specifically, the problem of finding an urban area parking spot, which can be distinguished into on-street and off-street parking, has been examined.

Kanza et al. [6] calculate routes to a given destination on a probabilistic graph, maximizing the certainty of visiting relevant points of interest. Hua and Pei [4] study routing under uncertain travel time. They either bound probability or duration and optimize for the other. In contrast, we consider a multi-objective problem where the probability is a hard constraint. Moreover, their algorithms assume a specified destination which does not apply in parking search, thus we cannot apply their algorithms to our problem.

Probabilistic routing is usually modeled as a graph of resources, each of which can either be available or not [5,6,11]. Kanza et al. [6] and Jossé et al. [5] both abstract from the road network and span their graphs over resources only. Since we have a probability of parking success for each road, we must span our graph over the complete road network. While this is conceptually the same, it results in large graphs where backtracking, as used by Jossé et al. [5], is no longer suitable.

Kanza et al. [6] refer to uncertainty as the probability of a particular resource being relevant and available, similar to our per-street probabilities. As an answer to a *route-search query*, they suggest two complementary length-bounded and probability-bounded scenarios. We use the latter approach with our probability mass threshold. For an on-street parking search, a bounded-probability scenario makes sense since a parking spot can never be guaranteed completely and routes can potentially be infinite.

As mentioned, Jossé et al. [5] propose a resource graph model that they use to answer parking search queries. Their main focus lies on resource reappearance. In their model, the observed state of a resource decays over time, allowing consumed resources to reappear with a certain probability. They differentiate between long-term and short-term observations. Long-term observations correspond to our static probability model, while we do not model short-term observations because real reappearance data is not available. Further, our routes usually do not take too long to travel, making it unlikely that a parking spot will appear in a street already scanned.

Another technique which models search in uncertain environments are Markov Decision Processes (MDPs) [10]. Since we do not assume that parking spots reappear during the short search times we consider, the transition probabilities in our problem depend on the history of traversed streets. Because the problem is not memoryless, it does not allow modeling those probabilities as MDPs.

3. Model

Our goal is to optimize the time spent for the whole parking search process. To do so, we model a road network as a directed graph whose nodes represent crossings and whose edges represent streets. Further, each edge e is augmented with information about its (walking) distance $d(e)$ to the destination, its time $t(e)$ to traverse the edge, and the probability $p(e)$ of finding a parking spot.

The search process starts at a crossing in the graph, given by a specified node v_0. For simplicity, this node is also the desired destination, since we assume that the driver has already reached the destination and will now start looking for a parking spot from there. Our proposed algorithms can also work with other destinations without any modifications. The parking route is represented as a path P on the graph and is considered successful if the probability of *not* finding a parking spot is at most ε.

The overall cost of a successful path is a convex combination with a parameter λ of, on the one hand, the time spent not finding a parking place (the *driving duration*) and, on the other hand, the distance to the destination v_0 (the *walking distance*). We do not

let probabilities reappear – meaning an edge can contribute a positive probability of finding a parking spot at most once, since we assume that a driver will most likely not want to drive through the same street twice. Further, if we would let probabilities reappear, this would lead to an unrealistic scenario where it is preferable to constantly drive through the same street that has a high chance of finding a parking spot.

Since a driver usually scans opposing lanes of a single street at once, we further introduce a function D that maps edges of the graph to sets of edges that share a single probability. If we traverse an edge (u, v), the probabilities of all edges in $D((u, v))$ disappear as well. Normally, for an edge (u, v), $D((u, v))$ would consist of at most (u, v) and (v, u). If, however, opposing lanes were separated, only choosing $D((u, v)) = \{(u, v)\}$ would make sense. In the following definition, we make use of D in propositions of the form $\forall k < j : e_j \notin D(e_k)$, which say that we only use indices that did not contribute any probability yet.

Definition 1 (Minimal Parking Spot Search)**.**
Instance: Directed graph $G = (V, E)$, time function $t : E \to \mathbb{R}_{\geqslant 0}$, distance function $d : E \to \mathbb{R}_{\geqslant 0}$, probability function $p : E \to [0, 1]$, specified vertex $v_0 \in V$, threshold $\varepsilon \in [0, 1]$, and value $\lambda \in [0, 1]$. Let $D : E \to \mathcal{P}(E)$ such that, for all $e \in E$, $e \in D(e)$.

Solution: Edge sequence $P = (e_1, e_2, \ldots, e_\ell) \in E^l$ with $\ell \leqslant |E|^2$ such that $e_1 = (v_0, v)$ for some $v \in V$ and $\forall\, i \in \{1, 2, \ldots, \ell-1\} \exists\, u, v, w \in V : e_i = (u, v) \wedge e_{i+1} = (v, w)$ and $\prod^{\ell}_{\substack{i=1: \\ \forall j<i\,:\, e_i \notin D(e_j)}} (1 - p(e_i)) \leqslant \varepsilon$.

Measure: Minimize the cost $c(P)$ defined as

$$
\begin{aligned}
&c(P) \\
&= \overbrace{\lambda \sum_{i=1}^{\ell} t(e_i) \cdot \left(\prod_{\substack{j=1: \\ \forall k<j\,:\, e_j \notin D(e_k)}}^{i-1} \big(1 - p(e_j)\big) \right)}^{\text{driving duration}} \\
&\quad + (1 - \lambda) \\
&\quad \cdot \underbrace{\sum_{\substack{i=1: \\ \forall k<i\,:\, e_i \notin D(e_k)}}^{\ell} p(e_i) \cdot d(e_i) \cdot \left(\prod_{\substack{j=1: \\ \forall k<j\,:\, e_j \notin D(e_k)}}^{i-1} \big(1 - p(e_j)\big) \right)}_{\text{walking distance}}.
\end{aligned}
$$

Note that our measure is equivalent to

$$
\begin{aligned}
&\sum_{\substack{i=1: \\ \forall k<i\,:\, e_i \notin D(e_k)}}^{l} p(e_i) \left(\prod_{\substack{j=1: \\ \forall k<j\,:\, e_j \notin D(e_k)}}^{i-1} \big(1 - p(e_j)\big) \right) \\
&\quad \cdot \left(\lambda \sum_{j=1}^{i} t(e_j) + (1 - \lambda) \cdot d(e_i) \right) \\
&\quad + \lambda \sum_{i=1}^{l} t(e_i) \cdot \prod_{\substack{j=1: \\ \forall k<j\,:\, e_j \notin D(e_k)}}^{l} \big(1 - p(e_j)\big),
\end{aligned}
$$

that is, a convex combination of the expected driving duration and the expected walking distance plus an error term, which models an optimistic extension of the route by a road which contributes zero time, zero distance, and a probability of one.

Given any route $P = (e_1, e_2, \ldots, e_\ell)$, we call the value $1 - \prod^{l}_{\substack{i=1: \\ \forall j<i\,:\, e_i \notin D(e_j)}} (1 - p(e_i))$ the *probability mass of* P. Note that this is the probability of having found a parking spot along the route, as it is the inverse probability of *not* having found a parking spot in each street.

4. Algorithms

We now introduce two algorithms for optimizing the problem of finding a parking spot quickly. The space of potential solutions for this problem spans a tree where each node represents a route. Each route begins with an outgoing road of the start node, and children in the tree extend their parent route by one possible edge each. Since each node has at least two children, the time of a brute-force search grows exponentially in the depth of this tree. In addition to that, our problem is NP-complete [2]. Thus, our algorithms only explore this solution tree partially, in order to make computations viable on large road networks.

The first algorithm we introduce is **Branch and Bound** (Algorithm 1). It finds a near-optimal solution and thus serves as baseline for a given probability mass threshold ε in our experiments. For the case of not having probability data available, we propose a **Heuristic Search** (Algorithm 2) algorithm that explores the solution tree shallowly and chooses a route that is best with respect to a certain heuristic.

Algorithm 1: B&B(*expands*)

```
queue ← queue with empty route;
best ← empty route;
while probability mass of best < 1 − ε do
    best ← best concatenated with outgoing edge
    e where p(e) / (c(best+e) − c(best)) is largest;
    if best.length > 50 then
        c(best) ← ∞;
        break;
for n = 1 to expands do
    route ← queue.pop();
    foreach outgoing edge from route do
        if c(route concatenated with edge) >
        c(best) then
            continue;
        if probability mass of route ⩾ 1 − ε then
            best ← route;
            continue;
        queue.push(route concatenated with edge);
return best;
```

Algorithm 2: BFSH(*expands*) with its heuristic objective h (Eq. (1))

```
queue ← queue with empty route;
for n = 1 to expands do
    route ← queue.pop();
    foreach outgoing edge from route do
        queue.push(route concatenated with edge);
return P ∈ queue where h(P) is largest;
```

4.1. Breadth-first branch and bound

Breadth-First Branch and Bound (**B&B**, Algorithm 1) has knowledge of per-street probabilities and the evaluation threshold ε. We follow the branch-and-bound paradigm by first obtaining an initial upper bound B and then exploring all branches of the solution tree that are still lower in cost than B. For the upper bound B, we greedily expand a single route from v_0. At each node, we choose an outgoing edge with the largest local benefit as the next segment. We then explore the whole solution tree, discarding any intermediate routes that exceed this bound. If a considered route reaches a probability mass of at least $1 - \varepsilon$ before getting pruned, it must be better than the previous bound. Thus, we update B with the cost of this route and proceed exploring the remaining solutions. We restrict the number of explored routes with an *expands* parameter. In comparison to restricting the depth of the search, this allows for consistent computation times, as it is independent of the local branching factors of the road network. We traverse the restricted solution tree in a *breadth-first search* manner to look at shorter routes first, with random ordering within each level.

Since our number of expands is fixed, a good initial greedy solution allows **B&B** to explore more edges, as worse solutions will be detected earlier. Thus, the quality of the overall result depends on the quality of the initial solution.

Note that in reality, a parking spot can never be guaranteed and routes found by **B&B** may not lead along a vacant parking spot. The algorithm can be modified to handle this scenario: if the returned route ends before we find a vacant parking spot, we restart the algorithm with the last visited node as new start node but the old distance values. Since it knows per-street probabilities, visited routes stay at zero probability. We use this approach for the experiment on inaccurate probabilities, where the algorithm sees disturbed probabilities and the returned route might not reach the probability mass threshold.

4.2. Breadth-first search with heuristic objective

For some cities, the data used for generating probabilities may not be available and only map data can be used. Therefore, we introduce **Breadth-First Search with Heuristic Objective** (**BFSH**, Algorithm 2), which computes routes using a classical *breadth-first search* approach, also limited by the number of *expands*. Since the algorithm does not access probabilities, it cannot evaluate the real cost of a route. In fact, **BFSH** only considers the cost of its route once it exhausted its number of expands. Then, it uses a heuristic h (Eq. (1)) in order to choose a good route from the explored solution tree. Thus, this algorithm does not optimize for the actual objective (the cost $c(P)$ of a route P) but a heuristic objective ($h(P)$).

We design $h(P)$ mainly based on the distances of the edges $d(e_i)$ in the path P to the desired destination. This makes the heuristic widely applicable. In our model, edges considered later in the algorithm tend to contribute less to the cost, since a parking spot most likely has already been found. Without the correct probabilities, we have to estimate this discount via

a function $s(i)$. We use a linear decreasing function for $s(i)$, but other models might apply as well.

Our heuristic is given by

$$h(P) = \sum_{\substack{i=0: \\ \forall j<i\,:\, e_i \notin D(e_j)}}^{|P|} \frac{s(i)}{d(e_i)}\,, \tag{1}$$

which should be maximized.

We tried to improve the heuristic using commonly available road attributes, such as information about nearby points of interest and the importance of a road, but none of these approaches yielded significant performance improvements. Incorporating more information, such as demographic data, could potentially yield better heuristics, but this data may not be as widely available.

Note that, contrary to **B&B**, a single run of **BFSH** may not lead to a route with a probability mass of at least $1 - \varepsilon$, as the algorithm has no access to the probabilities. However, if no parking spot has been found during one run, the algorithm can be restarted from the last visited node. In this case, the probabilities should not reappear.

4.3. G2

In our experiments, we also compare a greedy algorithm introduced by Jossé et al. [5], called **G2**, which has knowledge of the per-street probabilities as well. It greedily chooses the next edge that maximizes the probability per driving duration. That is, this algorithm is basically the greedy approach we use in our **B&B** in order to come up with a good initial solution but it chooses an edge e that maximizes $p(e)/t(e)$.

5. Data set

Algorithms, experiments, and study results are based on a data set consisting of 85,729 road segments and 43,564 nodes in Berlin, Germany. The raw data was collected by the automotive navigation systems producer TomTom as floating-car data. Anonymized GPS information results were analyzed and filtered in order to detect on-street parking events based on navigational destinations, driving speed, and behavior. The results were aggregated for each hour of day over a multitude of traces in order to find the number of parking searches and successes for each road segment. This allows us to calculate per-street probabilities for finding a parking spot successfully. The probabilities are added to an existing road data base containing common attributes such as *length*, *functional road class*, and *average speed* for each road segment. Unlike the other properties, the probabilities are broken up by hour of day and day of week to allow for time-dependent routing. Since the probability data were determined empirically, they include a human factor. For example, drivers might have rejected a vacant parking spot because of the small space, expensive parking fees, or other personal preference.

Because the data set contains a few unrealistic probabilities of 1.0 in streets with too few detected parking searches, we adjust the probabilities for a meaningful result. This is based on the number of observations for each edge. For this, we use the lower bound of the Agresti–Coull confidence interval [1] with a confidence level of 95 %. That means we can be 95 % certain that a probability value for a road segment lies within an interval $[a, b]$. We assign a to the road as the new probability. This results in pessimistic estimates of parking probabilities.

Since probabilities are not available for all cities and roads, it would be interesting to predict them based on other attributes. And indeed there is a correlation between the probability and some attributes. Unfortunately, every correlation we found can be explained by the length of a road segment. In more detail, when normalizing each road by its length, there is no significant correlation among the attributes. This insight does not allow us to predict a probability but leads us to the conclusion that the absolute probability of a road depends on the length of the segment. Thus, for some experiments it is necessary to consider the *probability density* of a road: concerning a road e, we define its probability

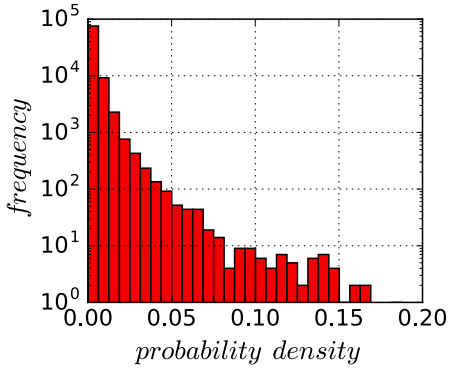
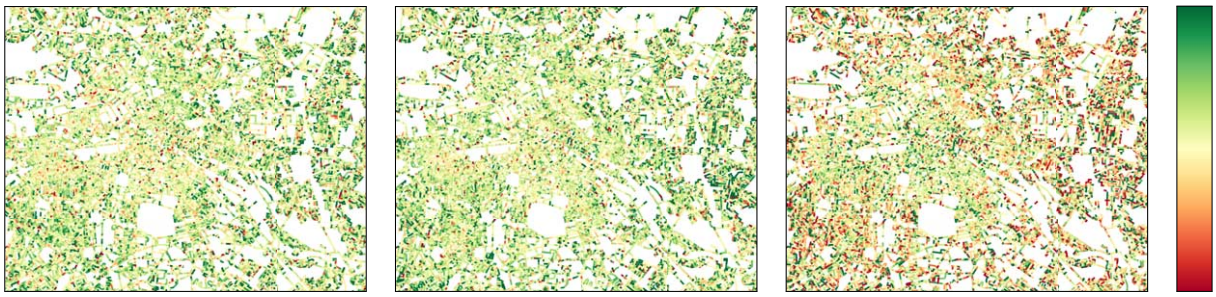


Fig. 1. The *probability density* is a measure for the parking probability per meter. For our dataset, we assume an underlying Pareto II distribution. (Data from Mon–Sun, 9 a.m.–4 p.m.)



(a) Morning, 6 a.m. to 2 p.m. **(b)** Evening, 2 p.m. to 9 p.m. **(c)** Night, 9 p.m. to 6 a.m.

Fig. 2. We compute the probability densities of all roads at different times of day. The three heat maps show differences with respect to the per-street averages over the whole day. Containing 90 % of values, we map the interval $(-0.003, +0.003)$ from red (worse than average) to green (better than average). One can see that parking becomes harder at night, especially in residential areas, which can also be observed in our experiments. (All data for Mon–Sun.)

density, given its probability $p(e)$ and length $\ell(e)$, to be $1 - (1 - p(e))^{\frac{1}{\ell(e)}}$, which resembles the probability of a unit-length part of the road. The collected probability mass stays the same, no matter whether we observe the whole road at once or each unit length separately. Therefore, using densities counteracts the arbitrary splitting of roads into segments in the map.

The probability mass is approximately Pareto distributed, as shown in Fig. 1. The fitted distribution (with parameters $\sigma = 1.3445$, $\mu = 0.0000$, and $\xi = 0.0022$) is the basis for the experiments on inaccurate probabilities, which we will introduce later. We further compare the probability densities over Berlin at different time spans with the densities averaged across the whole day in Fig. 2. We did not find a dramatic difference between weekdays and weekends.

6. Experiments

We conduct experiments in three scenarios, which we will describe and interpret. In all of them, we compare our **B&B**, our **BFSH**, and **G2**. We do not consider computation time in our evaluation but the solution qualities of the algorithms because all three algorithms need less than one second to compute a route. A query time in this order of magnitude does not affect applicability in practice.

In contrast to the problem formalization, we weight the two cost terms slightly differently in order to make interpretation easier: while in the formalization the *driving duration* and the *walking distance* are weighted with factors λ and $1-\lambda$, respectively, we first divide the walking distance by a typical walking speed of 1.4 $\frac{\text{m}}{\text{s}}$ [9] and then add up both measures with equal weight. Therefore, $c(P)$ refers to the total duration in seconds of finding a parking spot and walking to the destination.

6.1. Setup

We consider three scenarios and analyze how well the algorithms from Section 4 perform. That is, we look at the costs of the algorithms when starting from various nodes in the graph. First, we consider the performance during different times of a day. Second, we look at different weightings of the driving duration and walking distance, i.e., different values of λ. Last, we add noise to the edge probabilities in different intensities.

In each scenario, we let the algorithms run from the same set of 1000 randomly sampled start nodes v_0. Each such node is chosen with a probability proportional to the amount of parking searches in the time bin as of our dataset. If not stated otherwise, the time bin we use for the simulation is Monday to Friday from 9 a.m. to 5 p.m.

For all experiments, our probability mass threshold ε is 0.05. That is, we say that an algorithm produced a feasible solution when the probability of having found a parking spot is at least 95 %. Whenever an algorithm reaches this threshold, we stop the run and look at the cost of the produced route. The decision of when an algorithm should stop is *not* done by the algorithm itself but by the testing framework.[4] This evaluation is always done with respect to our original (unnoisy) probability data. Note that this entails that **BFSH** as well as

[4] This resembles the real-world equivalent of the driver telling the algorithm when to stop proposing new routes.

any of the three algorithms in the noisy setting may not produce a feasible solution during a single run. If this is the case, we restart the algorithms from the current node but do not reset the probabilities.

In order to prevent infinite loops when an algorithm drives through cycles of streets with no probabilities, we limit the route length to 100 edges. This means that a run of an algorithm can fail if the proposed routes exceed this limit. If an algorithm fails at least 5 % of the runs, we declare it as failing in a certain scenario.

In the first and third scenario, we choose $\lambda = \frac{1}{2}$, i.e., we weight driving duration and walking distance equally.

For **B&B**, the parameter *expands* is 10,000, while for **BFSH** it is 1000.[5] For both algorithms, we did not find a significant improvement in cost above those values. In other applications, we suggest experimentally determining the number of expands, as they may vary based on the road network. Further, we use a falloff function of $s(i) = 1 - \frac{i}{20}$ for **BFSH**, which we determined via a parameter search. **G2** does not have any free parameters.

For each scenario, we provide the experimental results as box plots of the *absolute* cost c (Fig. 3) as defined in Definition 1 and the *relative* cost c_r (Fig. 4) with respect to **B&B**. This means that for Fig. 4, **B&B** is only depicted for the noisy scenario, since we use its cost in the unnoisy scenario as baseline.

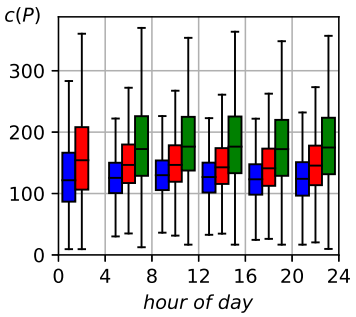
[5]The difference in these values stems from **BFSH** usually being restarted whereas **B&B** is not.

6.2. Time of day

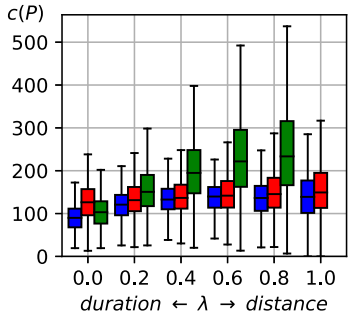
In Fig. 3(a), we compare the three algorithms against each other in terms of cost throughout the day. Since some roads have a 0 % parking probability at some times due to the lack of data for that time point, we merged the data into four-hour intervals.

We see that all algorithms perform worst during the interval from 12 a.m. to 4 a.m. The variances of **B&B** and **BFSH** are largest, and **G2** even failed during this interval. This is presumably due to the very low parking probabilities during this time of day (cf. Fig. 2(c) for an impression). The failure of **G2** shows that its myopic behavior is not preferable when the parking probabilities are low. Considering multiple alternative routes seems to be a clear advantage here. In the other intervals, the performance per algorithm is almost the same. Thus, all algorithms seem to be rather stable with respect to the hour of day.

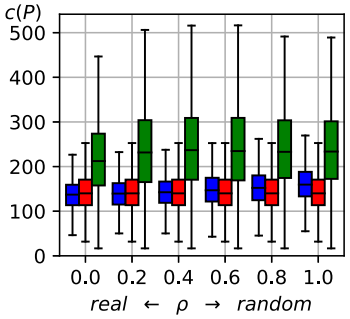
Comparing the algorithms relatively when considering Fig. 4(a), we see that **B&B** basically always outperforms **G2**. This is due to **B&B** using a greedy approach very similar to **G2** before it starts pruning. In contrast to that, **BFSH** sometimes outperforms **B&B** although it only does so outside its middle 50 %. This may be a result of the restarts that **BFSH** uses: if it does not reach the probability threshold after its expands, it is restarted from its current location. This way, it can potentially consider routes that **B&B** will not consider.



(a) Average algorithm costs throughout the day. We only show data points where the algorithm finds a successful route in at least 95 % of all starting points.

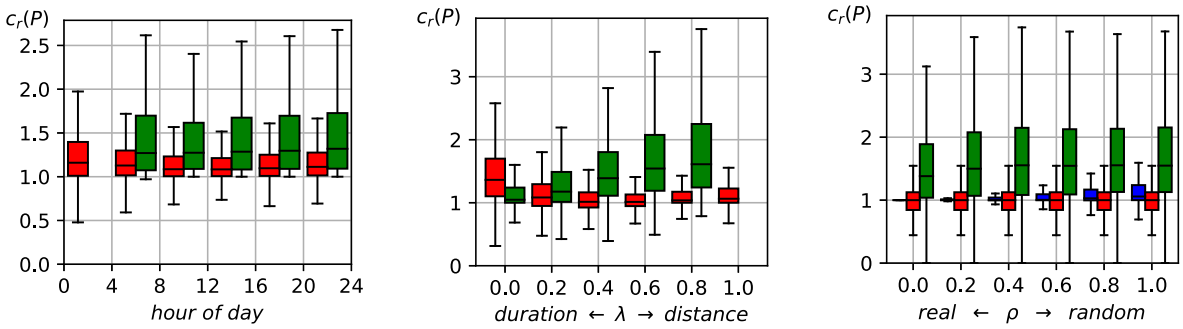


(b) Average algorithm costs at different weightings of driving duration and walking distance. Cost is doubled to match the other experiments at $\lambda = 0.5$.



(c) Average algorithm costs when operating on probability data that is disturbed to varying levels. **BFSH** is unaffected.

Fig. 3. Algorithm performance in the three experiments we conducted as described in Section 6.1. Displayed are the algorithms (from left to right): **B&B** (blue), **BFSH** (red), and **G2** (green) by Jossé et al. [5]. The black line inside each colored box depicts the median. The boxes themselves indicate the interquartile range (middle 50 %). The whiskers limit 95 % of all values.



(a) Algorithm costs throughout the day, relative to the baseline. We only show boxes where the algorithm finds a successful route in at least 95 % of all starting points.

(b) Algorithm cost at different weightings of driving duration and walking distance, relative to the baseline.

(c) Algorithm cost using probabilities with different levels of accuracy, relative to the baseline (at $\rho = 0.0$).

Fig. 4. The data from Fig. 3 as relative performance. Figures 4(a) and 4(b) display (from left to right) **BFSH** (red), and **G2** (green) and Fig. 4(c) displays (from left to right) **B&B** (blue), **BFSH** (red), and **G2** (green). Per run, the cost of each algorithm was divided by the cost of our (unnoisy) baseline **B&B**.

6.3. Cost weighting

Until now, we have weighted driving duration and walking distance equally. Because walking speeds and user preferences may differ, this weighting is not universal. In order to investigate the influence of the weighting parameters on the algorithms, we ran our experiments with a range of different weightings. For $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$, we ran the algorithms with the same starting points. The new weighted cost is defined as $c(P) = 2(\lambda \cdot \text{walking\_distance} + (1 - \lambda) \cdot \text{driving\_duration})$. For a correct interpretation, it is important to note that **B&B** *did* have access to this cost function, including the weighting factor λ.

Figure 3(b) shows that all algorithms perform better when the driving duration is weighted high, since it is faster to pass segments with high probabilities than walking back from them. However, for **B&B** and **BFSH**, mostly the variance increases, whereas **G2** gets considerably worse in general – even to the point where it fails (when we only consider distance). This makes sense, as **G2** chooses those edges greedily that have a good ratio of parking probability to driving duration. That is, it completely ignores walking distance.

In general, we see that **BFSH** mostly performs quite similarly to **B&B**, the single exception being when we only consider driving duration. This is due to the heuristic of **BFSH**, which only optimizes for the walking distance, as equation (1) only uses $d(e)$ and not $t(e)$.

Overall, we conclude that it is better to focus on the walking distance than on the driving duration, as indicated by the different behavior of **BFSH** and **G2**.

6.4. Impact of inaccurate probabilities

Despite a high number of parking observations, we can assume that computed success probabilities do not fully correspond to real-life circumstances. Hence, we need to assess the impact of inaccurate probability values. We do this by applying an interpolated noise model. When simulating noise, all algorithms work with gradually mutated probabilities, while their results continue to be evaluated on the original setting with ground-truth data provided by TomTom. This means that an algorithm requesting the costs of its next step would obtain a noisy response and act potentially less accurate. A probability-agnostic algorithm, however, would always perform the same, no matter which noise setting. A noise parameter ρ from the interval $[0, 1]$ determines the level of accuracy, with 0 implying unaltered probabilities and 1 fully applied noise. We have conducted simulations on various noise distributions, but in this experiment, we focus on an interpolated Pareto II distribution. For this model, we take random samples from a Pareto distribution fitted on the probability densities of all roads on the map. The resulting density per road segment with applied noise is the sum of its original density weighted by $1 - \rho$ and the sampled density weighted by ρ.

Figure 3(c) compares the costs of our algorithms on noise levels within the interval of [0, 1]. As expected, the performance of **B&B** and **G2** decreases with a rising noise level, as the observed data deviates more strongly from the actual data. However, this decline in quality is not very large, and the variance of each algorithm remains almost the same. That G2 does not fail in at least 95 % of the runs for all noise levels may be attributed to many fairly easy runs that have many streets with acceptable probability. In this case, it does not matter drastically which road segment to choose next.

An interesting observation is that **G2** regularly outperforms **B&B**. This behavior may result from an incorrect evaluation of the greedy baseline of **B&B** for pruning (similar to **G2**) due to the noise. Thus, **B&B** continues to prune routes based on wrong information, which results in worse routes. Thus, for certain runs, it would make sense to not try to improve the initial solution. However, the middle 50 % of **B&B** are still far better than **G2**.

As the performance of **B&B** and **BFSH** are quite similar, it is interesting to compare them. Note that **BFSH** does not change during different noise levels, since it is noise-oblivious. We observe that **B&B** is better than **BFSH** up to a noise level of $\rho = 0.6$. Thus, granting an algorithm access to probability data (even if they are inaccurate) seems to be often better than using no probability data at all.

7. Field study

Experimenting and analyzing in theory can offer important insights, but it still examines just a model of a real-world problem. In order to find out whether an algorithm is in fact helpful regarding the parking spot search, we conducted a field study. Within this study, we compared the introduced algorithms **B&B** and **BFSH** and, additionally, the human intuition.

When driving under the guidance of the two algorithms, new aspects occur. First, turning in a different direction costs a different amount of time on average. Second, in some situations, maneuvers are forbidden by the road traffic regulations. Both of these aspects had to be considered for the resulting routes to be feasible in their execution. For the study, we thus added turning penalties to the cost function with respect to the angle of the turning maneuver. The additional costs are based on the turning penalty model introduced by Luxen and Vetter [8].

As before, we sampled the destinations proportional to the amount of parking searches. The respective hour, its prior, and its subsequent hour of day were aggregated in order to have a larger amount of data to sample on. Due to reasons of feasibility, we took the nearest few samples to the current location of the car. Different districts of Berlin were selected in order to ensure that the results are representative for the whole city.

When drivers had already searched for parking spots in one situation, it would not be legitimate to let them drive in another situation within the same area. This is because they could have knowledge of available parking spots from the previous situation, which they would usually not have if this were their first situation. Therefore, no sample is closer than 1,300 meters to another, in the case of one driver.[6]

A *destination* is represented by an edge of the road graph. More specifically, the crossing v the edge (u, v) is pointing at is the start of a *run* as well as the desired destination (a run is exactly one search for a parking spot). Thus, it is considered as the point where the driver would have to walk after finding a parking spot. At every destination, three runs are performed starting at this destination: one guided by human intuition and one under navigation of each of the two algorithms. A run lasts for five minutes or until five parking spots are found. At each destination, the three runs are performed by the same driver. All drivers first search intuitively to guarantee that they do not have any knowledge from previous runs. Then a run is performed under the guidance of **B&B** and one using **BFSH**. However, a run is not performed if the suggested route is exactly the same as in one of the previous runs. Instead, the run is noted as equivalent to the previous one. To obtain comparable results, every run at a destination is performed by the same driver.

Each run was recorded via GPS using the Android app Geo Tracker.[7] Every time the driver found a parking spot that was big enough to park an average car, its GPS coordinates and the current time stamp were recorded. The driver then continued driving without delay. The total driving duration for a parking spot was then calculated from the time stamps as the time it took the driver to find the respective spot. The walking distance was calculated from the GPS coordinates as the beeline distance from the spot to the origin of the search.

[6] 1,300 m is an estimate of five minutes driving distance in the city of Berlin.

[7] geo-tracker.org

7.1. Discussion

In this field study, seven students participated as volunteers. 39 destinations were selected, two of which had to be aborted due to construction work and one due to outdated map data. 36 situations were used for the presentation of results and their discussion. Within these 36 situations, 87 runs were performed, and 21 times an algorithm suggested exactly the same route which already has been performed before. The field study was conducted over four days from a Monday to a Thursday between 10 a.m. and 8 p.m., each day.

Some situations turned out to be very easy regarding finding a parking spot. Therefore an *easy situation* is defined as such if both algorithms and the human intuition found five parking spots within five minutes. All others are defined as *difficult situations*. Since the purpose of testing **B&B** and **BFSH** was to find out whether they can optimize the search for a parking spot, it is not necessarily sensible to consider situations where parking is not a problem in the first place. Therefore, the following analysis focuses on *difficult situations*. 15 out of the total 36 situations were considered difficult according to this definition.

The comparison considers two aspects of each run: the number of parking spots (Fig. 5) and the average cost of all parking spots (Fig. 6). The former describes the total number of parking spots found by the respective algorithm, the latter the cost of all parking spots found. This cost again consists of the driving duration and the walking distance converted into time. For this, we use the Euclidean distance to the destination as an estimate and divide it by the typical walking speed of 1.4 $\frac{\text{m}}{\text{s}}$ [9].

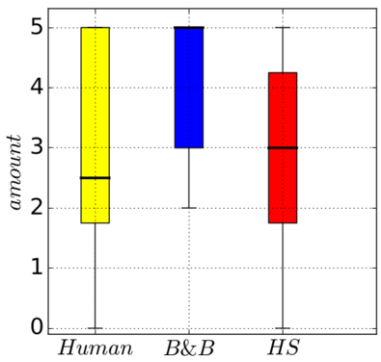


Fig. 5. The number of parking spots found within a run. The plot shows 15 situations determined as difficult. Displayed are the **Human Intuition** (*Human*; in yellow; left) and the algorithms: **B&B** (*B&B*; in blue; center) and **BFSH** (*HS*; in red; right). Please refer to Fig. 4 for an explanation of the box plot.

Even though only one single parking spot is needed to park a car, finding and proposing more than one might offer advantages. For example, a greater selection of parking spots would give the driver the opportunity to choose a parking spot more suitable to their needs, especially in the matter of walking distance to the desired destination. Besides, there are other factors that could let a driver want to have a greater variety, for example, whether a parking spot is located in the shade or surrounded by puddles. As Fig. 5 shows, **B&B** finds 45 % more parking spots than the human intuition or **BFSH**.

It does not always make sense to just take the first parking spot that opens up. Another one might be found later that is closer to the destination and thus be lower in total cost by our definition. Thus, it might be sensible to examine the average cost of all parking spots found during a run. Figure 6 shows that the median average combined cost of both **B&B** and **BFSH** is lower than that of the human intuition. This suggests, that both algorithms are more forward-looking than human intuition. Furthermore, the standard deviation of the combined cost of **B&B** is lower than those of **BFSH** and human intuition. This may imply that **B&B** is more stable in its results and thus more predictable. The advantage of **B&B** both in median as well as in standard deviation would be even greater if all runs had been continued until 5 parking spots were found. Since **BFSH** and the human intuition in most cases came to a conclusion after 5 minutes, as can be seen in Fig. 5, they often only average over fewer than 5 values. Values with longer driving durations (and supposedly longer walking time) are not considered.

Figure 6 also allows us to compare the influence of driving duration and walking time on overall costs in a realistic scenario. It seems like for human intuition and **BFSH** the influence of both factors is comparably high in most cases. Only for **B&B** we note a difference with the median of walking time being distinctly lower than that of driving duration. Especially, it is lower than the walking time of **BFSH**. This might be due to scenarios where the run starts in an area with generally few parking spots. **B&B** is target-oriented enough to lead the driver to a close-by area with higher parking probabilities, while **BFSH** starts looking in a random direction, thus finding parking spots which might be further away. Also, for both algorithms and the human intuition the standard deviation of the walking time is lower than that of the driving duration. This might be attributed to walking distances being similar independent of where in a certain region a parking spot is

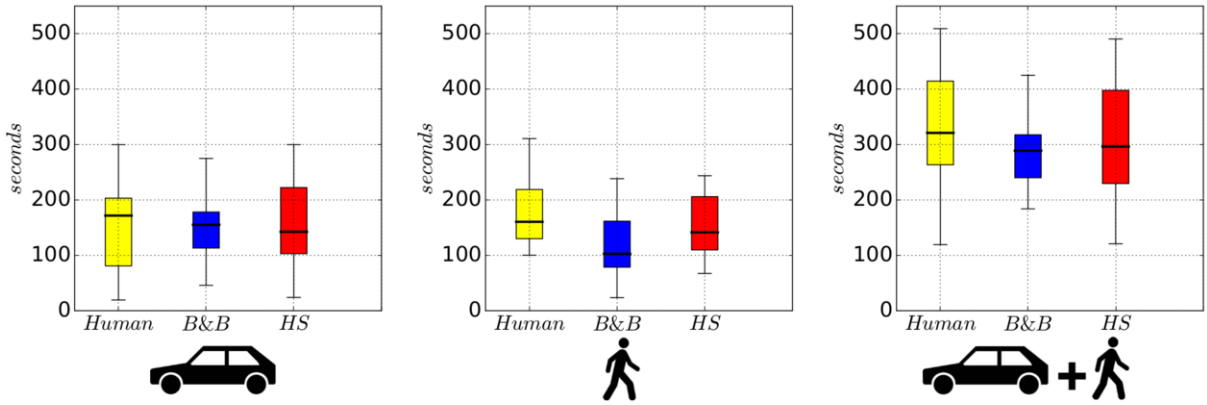


Fig. 6. The average cost of all parking spots found within a run in seconds, separated in driving duration (left), walking time (middle), and both summed up (right). The data presented is based on 15 situations determined as difficult. Displayed are (from left to right) the **Human Intuition** (*Human*; in yellow) and the algorithms **B&B** (*B&B*; in blue) and **BFSH** (*HS*; in red). Please refer to Fig. 4 for an explanation of the box plot.

found, i.e. the duration of the search does not affect the walking time as much as the driving duration.

We can also compare our experimental results to the ones from the field study. However, a quantitative comparison would not make sense, since our problem definition does not capture a lot of factors that affect driving duration in the real world, like traffic conditions and costs for waiting at intersections or traffic lights. Therefore, we only compare Fig. 3(c) and Fig. 5 qualitatively. We notice that the results of the field study look similar to one of the scenarios with little to no noise with respect to the relative positions of median and quantiles of **B&B** and **BFSH**. This suggests that the probabilities we derived from our data are not too noisy and can be used to achieve an advantage over **BFSH**.

8. Conclusion

State-of-the-art algorithms using probabilistic per-street parking data are able to find parking routes that are both shorter and closer to the destination than that of the human intuition. They offer a greater variety of parking spots and are generally able to save time. Our experiments show that even with inaccurate probabilities it is still possible to achieve benefits.

There are several possible improvements of our approach, which can be considered in the future. When searching for a parking spot, it is probably not best to only start searching after arriving at the desired destination. It rather is advisable to begin shortly before the arrival. In such a more likely scenario, an algorithm like **B&B** looking further ahead might yield even larger advantages over conventional parking search.

When many drivers (agents) use the same algorithm for finding parking spots, a challenging situation results in which the agents compete against one another for the available resources. One possible idea to diminish this effect would be to use a central system that optimizes for multiple routes at once such that all drivers are likely to find a parking spot. However, this approach gets infeasible when considering a great number of agents. Hence, it may make sense to let each agent act independently, using data from a server that caches different routes of other agents and adjusts the data to the current needs. These approaches all need access to a server, though. Thus, it may also make sense to have different algorithms stored on each device and choose one randomly when a system has no connection to the server. Alternatively, the algorithm itself could be randomized with respect do different parameters. Hence, multiple agents behave differently without the overhead of the server connection.

Another important next step involves improving the method of acquiring data. It would be best if some sort of live data could be used, like sensor data or information about major events that may block many parking spots, like sport games or music festivals. Another way to get better data could be to use the users themselves. For example, if a user found a parking spot and drives away after some time, this could be signaled to a server. Additionally, having information of where and what type of parking spots are in each street could also greatly improve the success rate, as the user could be told where exactly to look. Furthermore, parking spots that do not fit the need of the user could immediately be rejected by the algorithm.

Acknowledgements

We thank TomTom Development Germany GmbH, An den Treptowers 1, 12435 Berlin, Germany and its engineering team, including our contact people Steffen Wiesner and Peter Mieth. During our work, we received valuable feedback and support from them. They provided us with the collected probability data and additional road attributes, and we are grateful to have received early access to the dataset used in this paper.

References

[1] A. Agresti and B.A. Coull, Approximate is better than "exact" for interval estimation of binomial proportions, *The American Statistician* **52**(2) (1998), 119–126. ISSN 00031305. http://www.jstor.org/stable/2685469. doi:10.1080/00031305.1998.10480550.

[2] T. Arndt, D. Hafner, T. Kellermeier, S. Krogmann, A. Razmjou, M.S. Krejca, R. Rothenberger and T. Friedrich, Probabilistic routing for on-street parking search, in: *Proceedings of the 24th Annual European Symposium on Algorithms*, 2016, pp. 6:1–6:13. ISSN 1868-8969. ISBN 978-3-95977-015-6. http://drops.dagstuhl.de/opus/volltexte/2016/6348. doi:10.4230/LIPIcs.ESA.2016.6.

[3] S. Evenepoel, J. Van Ooteghem, S. Verbrugge, D. Colle and M. Pickavet, On-street smart parking networks at a fraction of their cost: Performance analysis of a sampling approach, *Transactions on Emerging Telecommunications Technologies* **25**(1) (2014), 136–149. doi:10.1002/ett.2776.

[4] M. Hua and J. Pei, Probabilistic path queries in road networks: Traffic uncertainty aware path selection, in: *Proceedings of the 13th International Conference on Extending Database Technology*, 2010, pp. 347–358. ISBN 978-1-60558-945-9. doi:10.1145/1739041.1739084.

[5] G. Jossé, K.A. Schmid and M. Schubert, Probabilistic resource route queries with reappearance, in: *Proceedings of the 18th International Conference on Extending Database Technology*, 2015, pp. 445–456. ISBN 978-3-89318-067-7. doi:10.5441/002/edbt.2015.39.

[6] Y. Kanza, E. Safra and Y. Sagiv, Route search over probabilistic geospatial data, in: *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases*, 2009, pp. 153–170. ISBN 978-3-642-02981-3. doi:10.1007/978-3-642-02982-0_12.

[7] C. Kulick, Parkplatzmangel nervt Stadtbewohner am meisten (in German), 2017. https://www.bitkom.org/Presse/Presseinformation/Parkplatzmangel-nervt-Stadtbewohner-am-meisten.html.

[8] D. Luxen and C. Vetter, Real-time routing with OpenStreetMap data, in: *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2011, pp. 513–516. ISBN 978-1-4503-1031-4. doi:10.1145/2093973.2094062.

[9] B.J. Mohler, W.B. Thompson, S.H. Creem-Regehr, H.L. Pick and W.H. Warren, Visual flow influences gait transition speed and preferred walking speed, *Experimental Brain Research* **181**(2) (2007), 221–228. ISSN 1432-1106. doi:10.1007/s00221-007-0917-0.

[10] M.L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons, 2014.

[11] E. Safra, Y. Kanza, N. Dolev, Y. Sagiv and Y. Doytsher, Computing a K-route over uncertain geographical data, in: *Proceedings of the 10th International Conference on Advances in Spatial and Temporal Databases*, 2007, pp. 276–293. http://dl.acm.org/citation.cfm?id=1784462.1784478. ISBN 978-3-540-73539-7. doi:10.1007/978-3-540-73540-3_16.

[12] D. Shoup, *The High Cost of Free Parking*, APA Planners Press, 2005. ISBN 1884829988.

[13] Toyota Motor Corp, Street parking availability estimation, Google Patents, 2014, US8797187B2. https://patents.google.com/patent/US8797187.

[14] E.I. Vlahogianni, K. Kepaptsoglou, V. Tsetsos and M.G. Karlaftis, A real-time parking prediction system for smart cities, *Journal of Intelligent Transportation Systems* **20**(2) (2016), 192–204. doi:10.1080/15472450.2015.1037955.