Probabilistic Routing for On-Street Parking Search

Tobias Arndt, Danijar Hafner, Thomas Kellermeier, Simon Krogmann, Armin Razmjou, Martin S. Krejca, Ralf Rothenberger, and Tobias Friedrich

Hasso Plattner Institute, Potsdam, Germany

Abstract

An estimated 30 % of urban traffic is caused by search for parking spots [8]. Suggesting routes along highly probable parking spots could reduce traffic. In this paper, we formalize parking search as a probabilistic problem on a road graph and show that it is NP-complete. We explore heuristics that optimize for the driving duration and the walking distance to the destination. Routes are constrained to reach a certain probability threshold of finding a spot. Empirically estimated probabilities of successful parking attempts are provided by TomTom on a per-street basis. We release these probabilities as a dataset of about 80,000 roads covering the Berlin area. This allows to evaluate parking search algorithms on a real road network with realistic probabilities for the first time. However, for many other areas, parking probabilities are not openly available. Because they are effortful to collect, we propose an algorithm that relies on conventional road attributes only. Our experiments show that this algorithm comes close to the baseline by a factor of 1.3 in our cost measure. This leads to the conclusion that conventional road attributes may be sufficient to compute reasonably good parking search routes.

1998 ACM Subject Classification I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases parking search, on-street parking, probabilistic routing, constrained optimization, dataset

Digital Object Identifier 10.4230/LIPIcs.ESA.2016.264

1 Introduction

Searching for a parking spot is expensive, time-consuming, and causes a significant amount of urban traffic: An estimated 30 % of traffic is due to people searching for parking spots [8]. Drivers could be assisted in their search by suggesting a route along streets with a high probability of yielding a vacant on-street parking spot. In this paper, we investigate the problem of generating these routes. While off-street parking options, such as car parks, are valid alternatives and have been explored, e.g., by Cassady and Kobza [2], we focus on *free of charge on-street parking*, which makes up a majority of the parking capacity in most cities [3].

Parking search is probabilistic by nature. Modeling the road network as a graph, each edge has a probability of having a vacant parking spot. Several works on probabilistic graph routing have been conducted as outlaid in section 2. Most relevant, Jossé, Schmid, and Schubert [5] recently proposed a probabilistic model for parking search that we build on.

In theory, parking routes can have an infinite length, since a parking spot can never be guaranteed. To evaluate our algorithms, however, we have to restrict routes to a finite length. In our formalization in section 3, we thus introduce a bound on the converse probability mass of routes, similar to previous work [5, 6]. Traversing the graph, at each edge the algorithm collects the probability of having found a parking spot. A route is considered successful if its collected probability mass reaches a certain threshold. See figure 1 for an example of a

successful route. We evaluate routes based on the two criteria *driving duration* and *walking distance* to the desired destination.

For generating good parking search routes, we propose two algorithms in section 4. The first one, **Branch and Bound**, is our baseline. Since it considers routes that reach a probability mass threshold constraint, it requires know ledge of per-street probabilities. For the case where those probabilities are not available, we propose a **Heuristic Search** algorithm that iteratively explores sub-routes scored by a heuristic. This approach comes close to the baseline by a factor of 1.3 in our cost.

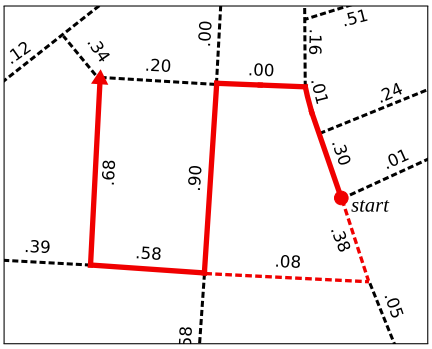


Figure 1 A simple route computed by our **Branch and Bound** algorithm (red line). While a greedy route (dashed red line) would collect more probability mass with the first two segments by leading South and West, the shown route reaches the 90 % edge slightly earlier and can collect the 58 % edge right afterward.

Along with our work, we release a *dataset of empirically estimated probabilities* of parking successes collected by TomTom. Those probabilities are given per-street and for each hour of day and day of week. The data was obtained from several million anonymously collected user records. In section 5, we describe the collection and preprocessing processes. We then explore characteristics of the dataset such as the difference of the probability distributions at day- and nighttime.

Based on the dataset, we evaluate our algorithms in section 6. First, we compare our **Branch and Bound** and **Heuristic Search** algorithms and a greedy algorithm proposed by Jossé et al. [5] at different *times of day*. Second, we compare different choices for *weighting the cost*. Since our cost is composed of two objectives, we can simulate different preferences. We show that finding a parking spot quickly tends to be easier than finding one that is close to the desired destination. Third, we assess the *impact of inaccurate probabilities* by considering our dataset as ground-truth and showing a disturbed version of it to the algorithms. We show that **Branch and Bound** still works well under a high level of noise but is outperformed by **Heuristic Search** later.

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 a parking spot in urban areas, 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 our scenario, after bounding the probability, we still have to optimize a multi-objective problem with *driving duration* and *walking distance*. Moreover, their algorithms assume a specified destination which does not exist in parking search, thus we can not apply their algorithms to our problem.

Probabilistic routing is usually modeled as a graph of resources that each can either be available or not [5–7]. 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 back-tracking, as used by Jossé et al. [5], is not suitable anymore.

Kanza et al. [6] refer to uncertainty as the probability of a particular resource being relevant and available, comparable 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 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.

3 Problem

3.1 Formalization

We now give a formal definition of our parking search scenario and prove that it is NP-complete. We model the road network as a directed graph whose edges are augmented with information about the distance of an edge to the destination, the time to traverse the edge, and the probability of finding a parking spot.

The search process starts at a crossing in the network, 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 now starts looking for a parking spot from there. Our proposed algorithms can 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 parameter λ of, on the one hand, the time spent not finding a parking place and, on the other hand, the distance to the destination v_0. Note that probabilities do not reappear, that is, an edge can contribute a positive probability of finding a parking spot at most once.

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 $D\big((u, v)\big)$ disappear as well. Normally, for an edge (u, v), $D\big((u, v)\big)$ would consist of at most (u, v) and (v, u). If, however, opposing lanes were separated, only choosing $D\big((u, v)\big) = \{(u, v)\}$ would make sense.

Definition 1 (Minimal Parking Spot Search (MPSS))

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

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

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

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

Note that our measure is equivalent to

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

The last term models an optimistic extension of the route by a road which contributes zero time, zero distance and a probability of one.

3.2 NP-Completeness

Let k-Minimal Parking Spot Search (k-MPSS) be the decision version of MPSS, i.e., the problem to decide if there is a feasible solution of cost at most k. We prove the NP-completeness of k-MPSS by a polynomial time reduction from Hamiltonian Path, i.e., the problem of finding a simple path in G which visits each node exactly once.

▶ **Theorem 1.** *k-MPSS is NP-complete.*

Proof. First of all, k-MPSS is in NP, since we can guess an edge sequence of size at most $|E|^2$ at random and it can be tested in polynomial time if the edge sequence is a feasible solution and if its cost is at most k.

Now we sketch how to reduce an instance of Hamiltonian Path to an instance of k-MPSS in polynomial time. Suppose we are given a directed graph $G = (V, E)$ as an input for Hamiltonian Path. We construct a new graph $G' = (V', E')$ with $V' = \{v_0\} \cup \{v^{\text{in}}, v^{\text{out}} \mid v \in V\}$ and $E' = \{(v_0, v^{\text{in}}) \mid v \in V\} \cup \{(v^{\text{in}}, v^{\text{out}}) \mid v \in V\} \cup \{(u^{\text{out}}, v^{\text{in}}) \mid (u, v) \in E\}$. We define $d(e) = 0$ for all $e \in E'$, $t(e) = 1$ for all $e \in E' \setminus \{(v_0, v^{\text{in}}) \mid v \in V\}$ and $t(e) = 0$ for all $e \in \{(v_0, v^{\text{in}}) \mid v \in V\}$. For all $e \in E'$, we choose $p(e) = \frac{1}{2}$ if $e = (v^{\text{in}}, v^{\text{out}})$ for some $v \in V$ and $p(e) = 0$ otherwise. Further, we define, for all $e \in E'$, $D(e) = \{e\}$, i.e., for two edges e_i and e_j, our proposition of $e_i \notin D(e_j)$ in the problem formalization simplifies to $e_i \neq e_j$. Finally, we choose $\lambda = 1$, $\varepsilon = 2^{-|V|}$ and $k = \sum_{i=1}^{2|V|-1} \left(\frac{1}{2}\right)^{\lfloor i/2 \rfloor}$. An example of such a reduction can be seen in Figure 2.

Suppose the graph G has a Hamiltonian path $(v_{i_1}, v_{i_2}, \ldots, v_{i_{|V|}})$. Then the edge sequence

$$\big((v_0, v_{i_1}^{\text{in}}), (v_{i_1}^{\text{in}}, v_{i_1}^{\text{out}}), (v_{i_1}^{\text{out}}, v_{i_2}^{\text{in}}), \ldots, (v_{i_{|V|}}^{\text{in}}, v_{i_{|V|}}^{\text{out}})\big)$$

is a feasible solution for k-MPSS. Since $(v_{i_1}, v_{i_2}, \ldots, v_{i_{|V|}})$ is a Hamiltonian path, each $(v^{\text{in}}, v^{\text{out}})$-edge is traversed exactly once with exactly one other edge in-between. Also, each edge is traversed at most once. The sequence consists of $2|V| \leq |E'|$ edges and amounts to

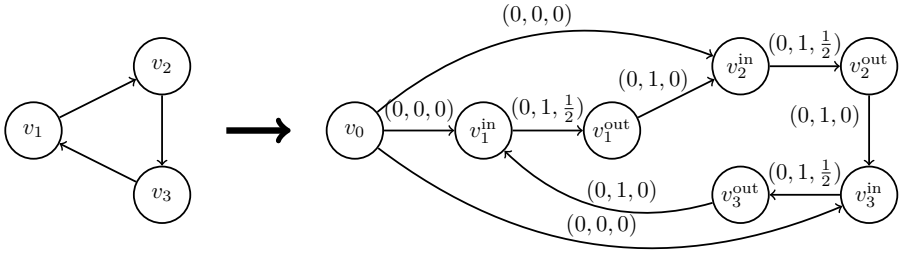


Figure 2 An exemplary reduction from Hamiltonian Path to k-MPSS. The graph on the left is transformed into the one on the right. The triple of an edge e in the right graph has the form $\big(d(e), t(e), p(e)\big)$.

$\prod_{\substack{i=0:\\ \forall j<i:\ e_i\neq e_j}}^{l} \big(1-p(e_i)\big) = 2^{-|V|} = \varepsilon$. The cost of the sequence is

$$\sum_{i=1}^{2|V|} t(e_i)\cdot\left(\prod_{\substack{j=1:\\ \forall k<j:\ e_j\neq e_k}}^{i-1}\big(1-p(e_j)\big)\right) = \sum_{i=2}^{2|V|}\left(\prod_{j=1}^{i-1}\big(1-p(e_j)\big)\right)$$
$$= \sum_{i=2}^{2|V|}\left(\frac{1}{2}\right)^{\lfloor (i-1)/2\rfloor} = \sum_{i=1}^{2|V|-1}\left(\frac{1}{2}\right)^{\lfloor i/2\rfloor} = k\ ,$$

since $t(e_1)=0$, $t(e_i)=1$ for all $i\geq 2$, $1-p(e_i)=\frac{1}{2}$ if i even and $1-p(e_i)=1$ if i odd.

To transform a solution $(e_1, e_2, \ldots, e_l)$ of k-MPSS into a solution of Hamiltonian Path, we simply take the nodes $(v_{i_1}, v_{i_2}, \ldots, v_{i_{|V|}})$ according to the order in which their corresponding $(v^{\text{in}}, v^{\text{out}})$-edges are traversed.

Now suppose we have a solution $(e_1, e_2, \ldots, e_l)$ of k-MPSS. To achieve

$$\prod_{\substack{i=0:\\ \forall j<i:\ e_i\neq e_j}}^{l}\big(1-p(e_i)\big) \leq 2^{-|V|}\,,$$

all $(v^{\text{in}}, v^{\text{out}})$-edges have to be visited at least once. Due to the construction of G', every edge sequence has to alternate between $(v^{\text{in}}, v^{\text{out}})$-edges and $(u^{\text{out}}, v^{\text{in}})$-edges. The factor in the cost term only changes if we visit a new $(v^{\text{in}}, v^{\text{out}})$-edge. If we visit any other edge or an edge we already visited, the same factor as before is added to the cost. That means, visiting e_i with $i>1$ adds $(1/2)^{\left|\{(v^{\text{in}},v^{\text{out}})|v\in V\}\cap\{e_1,e_2,\ldots,e_{i-1}\}\right|}$ to the cost. Since each $(v^{\text{in}}, v^{\text{out}})$-edge has to be visited at least once and the edge sequence has to be alternating, starting at some edge (v_0, u^{in}) with cost 0, the cost of every feasible edge sequence is at least k. Furthermore, to reach a cost of at most k, each $(v^{\text{in}}, v^{\text{out}})$-edge can be visited at most once, i.e., after transforming the solution, we get a path in G which visits each node exactly once. ◀

4 Algorithms

We now introduce two algorithms for optimizing the problem defined in section 3. The space of potential solutions for this problem forms a tree where each node holds 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. Because of the number of outgoing edges at each node, the

Algorithm 1: Branch and Bound(*expands*)

```
1  queue ← queue with empty route;
2  best ← empty route;
3  while probability mass of best < 1 − ε do
4      best ← best concatenated with outgoing edge e where p(e) / (c(best+e) − c(best)) is largest;
5      if best.length > 50 then
6          c(best) ← ∞;
7          break;
8  for n = 1 to expands do
9      route ← queue.pop();
10     foreach outgoing edge from route do
11         if c(route concatenated with edge) > c(best) then
12             continue;
13         if probability mass of route ≥ 1 − ε then
14             best ← route;
15             continue;
16         queue.push(route concatenated with edge);
17 return best;
```

time of brute-force search in this tree grows exponentially in depth. As we already showed, the problem is NP-complete, so our algorithms only explore this *solution tree* partially, in order to make computations viable on large road networks.

Our **Branch and Bound** algorithm serves as *baseline* for a given probability mass threshold ε. For the case of not having probability data available, we propose a **Heuristic Search** algorithm that explores the solution tree shallowly to compute sub-routes. We iterate this routine to obtain routes of theoretically infinite length. Since we cannot compute the cost of a route without probabilities, we have to make estimations. For this, we mention heuristics varying in their required knowledge of road attributes.

4.1 Branch and Bound

Branch and Bound, as shown in algorithm 1, has knowledge of per-street probabilities and the evaluation threshold ε. As an outline, we follow the branch and bound paradigm and obtain an initial upper bound B on the cost of a best route from v_0. We then explore the solution tree by pruning intermediate routes that exceed this bound. If a considered route reaches ε 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.[1]

For the initial upper bound, we greedily expand a single route from v_0. At each node, we choose an outgoing edge e with the largest term $\frac{p(e)}{c(P+e)-c(P)}$ as the next segment. Eventually, this route reaches ε. The cost of this serves as an initial upper bound $B := c(P)$ on the cost of the best route from v_0 that satisfies the threshold constraint.

[1] Jossé et al. [5] also proposed an algorithm based on the branch and bound paradigm. They suggest an expensive forward estimation of the remaining cost of intermediate routes. We found this to be ineffective compared to the increased amount of routes that can be considered otherwise.

We then explore the whole solution tree discarding any intermediate route P' if $c(P') \geq B$. 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. We use this algorithm as baseline in the experiments in section 6.

Note that in reality a parking spot can never be guaranteed and routes found by **Branch and Bound** may not lead along a vacant parking spot. The algorithm can be modified to handle this scenario: If the returned route ends before we found a vacant parking spot, we restart the algorithm with the last visited node as new start node. Since it knows per-street probabilities, visited routes stay at zero probability. We use this approach for the experiment on inaccurate probabilities in section 6.3, where the algorithm sees disturbed probabilities and the returned route might not reach the probability mass threshold.

4.2 Heuristic Search

For some cities, the data used for generating probabilities may not be available and only map data can be used. Therefore, we introduce **Heuristic Search** (algorithm 2) which computes sub-routes using *breadth-first search*, also limited by the number of *expands*. Since the algorithm does not access probabilities, it cannot evaluate the real cost for a route. In order to choose a good route from the explored solution tree, it uses a heuristic $h(P)$. Whenever the current route is exhausted, the algorithm restarts to compute the next sub-route. However, it memorizes all visited edges.

Algorithm 2: Heuristic Search(*expands*)

```
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;
```

We design $h(P)$ mainly based on the distances of the edges in P to the desired destination. This makes the heuristic widely applicable. In our formalization, later edges tend to contribute less to the cost since a parking spot likely is already found. Without the correct probabilities, we have to estimate this discount function $s(i)$. Our heuristic is given by

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

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 them yielded significant performance improvements. Although there was some correlation to the probability, nearly all of it could be explained by a shared correlation to the road length. Correcting this by using the probability density as defined in section 5 yielded no significant correlations. Incorporating more information, such as demographic data, could potentially yield better heuristics, but this data may not be as widely available.

5 Dataset

Complementary to the specification of a theoretical model, we evaluate realistic scenarios on the central area of Berlin, Germany. In the following section, we describe this dataset and its collection process conducted by TomTom[2]. While some of the provided information could be extracted from public sources, we did not identify any party that made a comparable amount of on-street parking measures available or used such data for research on simulated parking scenarios. Our dataset will be made available at `https://hpi.de/friedrich/research/parking/`.

The dataset contains a road network reflecting the graph described in section 3. Each edge contains data about the *nodes it leads from and to*, its *opposite edge*, the *parking probability*, and various other properties like the *length*, *functional road class*, and *average speed*. Unlike the other properties, the probabilities are broken up by hour of day and day of week. Since we determined the probability data empirically, they include a human factor. For example, drivers might have rejected a vacant parking spot because of small space or expensive parking fees. We interpret our experiments in section 6 accordingly.

The raw data was collected by TomTom as floating-car data: Anonymized GPS positioning-information was analyzed and filtered to detect on-street parking events based on navigational destinations, driving speed, and behavior. The results were aggregated over a multitude of traces to find the number of parking searches and successes for each road segment. We further adjust the probabilities for each road, using the lower bound of the Agresti-Coull confidence interval [1] with a confidence level of 95 %. This results in pessimistic estimates of parking probabilities, based on the number of observations for each edge. Thus, we avoid unrealistic probabilities of 1.0 that might be observed in streets with too few detected parking searches for a meaningful result.

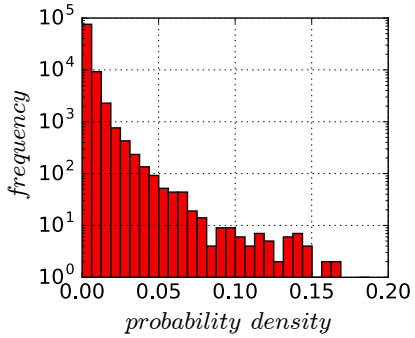
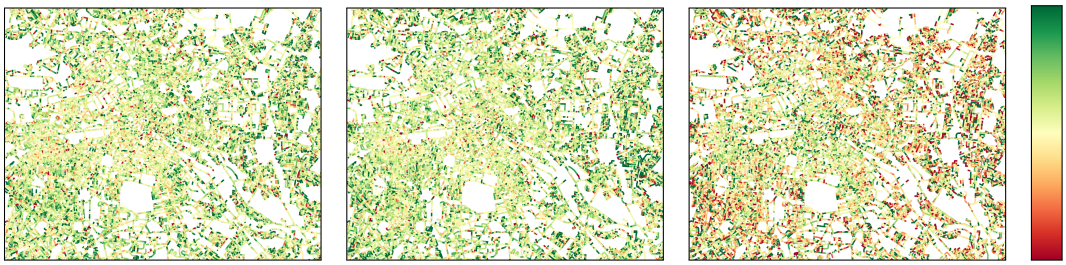


Figure 3 The *probability density* is a measure for the parking probability per meter. For our dataset, we assume an underlying Pareto II distribution. (Mon–Sun, 9 a.m.–4 p.m.)

From the probability $p(e)$ and length $l(e)$ of a road e, we compute a *probability density* $1 - \big(1 - p(e)\big)^{\frac{1}{l(e)}}$ that 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. This is necessary, because the probability correlates strongly with the length of the road, because long roads can simply fit more parking spots. Therefore, we can counteract arbitrary splitting of roads into segments in the map by using densities.

We found the probability mass to be Pareto-distributed as shown in figure 3. The fitted distribution with parameters $\sigma = 1.3445$, $\mu = 0.0000$, and $\xi = 0.0022$ is the basis for our experiments on inaccurate probabilities in section 6. We further compare the probability density at different time spans with the density averaged across the whole day in figure 4. Between weekdays and weekends, we did not find a prominent difference. We also tried to reconstruct the probabilities from other attributes of that road, using a statistical model, which did not yield meaningful results. This, however, emphasizes the uniqueness of the released dataset.

[2] TomTom Development Germany GmbH, An den Treptowers 1, 12435 Berlin, Germany



(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.

Figure 4 We compute the probability densities of all roads at different times of day. The three heat maps show differences 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 in section 6. (All data for Mon–Sun; best viewed in color.)

6 Experiments

We conduct three experiments that we will describe and interpret in this section. In all of them, we compare our **Branch and Bound**, our **Heuristic Search**, and the **G2** algorithm proposed by Jossé et al. [5], which greedily chooses the next edge that maximizes the probability per cost. We do not consider computation time in our evaluation because all three algorithms take less than one second to compute. A query time in this order of magnitude does not affect applicability in practice.

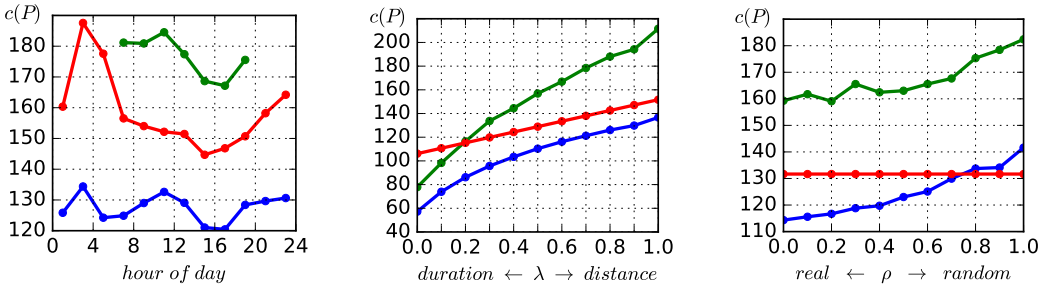
In contrast to the problem formalization in section 3, we weight the two cost terms slightly differently to make interpretation easier. While in the formalization, *driving duration* and *walking distance* are weighted with factors λ and $1 - \lambda$, we first divide the walking distance by a typical walking speed of $1.4\,\frac{\mathrm{m}}{\mathrm{s}}$ and then add up both measures. Therefore, $c(P)$ refers to the total duration for finding a parking spot in this section.

In all the following experiments, we choose probability mass threshold $\varepsilon = 0.05$. Thus a route is considered successful if it leads along a vacant parking spot with a probability of at least 95 %. We sample 1000 start nodes v_0, 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 **Branch and Bound**, the *expands* parameter is 10000 while for **Heuristic Search** it is 1000. For both algorithms, we did not find a significant improvement in cost above those values. In other applications, we suggest to experimentally determine the number of expands as they may vary based on the road network. Further, we use a fall off function of $s(i) = 1 - \frac{i}{20}$ for **Heuristic Search** that we determined through parameter search. **G2** does not have any free parameters.

To prevent infinite loops when algorithms drive through cycles of streets with no probabilities, we limited the route length to 100 edges. We do not include routes that exceed this length. If an algorithm does not reach the probability mass threshold ε in at least 95 % of all sampled starting positions v_0 we declare it as failing in a certain scenario.

The box plots in figure 6 provide statistical information for the costs of the algorithms used in the line diagrams in figure 5. The whiskers limit 95 % of all values while the box represents the first and the third quartiles with the central black line showing the median.



(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 v_0.

(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. Our heuristic search algorithm is unaffected.

Figure 5 Algorithm performance in the three experiments we conducted. Averaged over 1000 sampled start nodes v_0. From bottom to top: **Branch and Bound**, **Heuristic Search**, and **G2** by Jossé et al. [5].

6.1 Time of Day

In 5a we compare the three algorithms against each other in cost throughout the day. To avoid the aforementioned infinite loops, we merged the data into two-hour blocks. Nevertheless, **G2** achieved ε only during rush hours in more than 95 % of the runs. **Heuristic Search** and **Branch and Bound** both do experience decreasing costs in the early morning and the afternoon. Besides that, **Branch and Bound** is relatively consistent while **Heuristic Search** has a significant peak between 2 and 4 a.m., being 30 % more expensive than at its optimal time. Over the whole day, the baseline stays ahead of our heuristic approach by a factor of 1.3 on average.

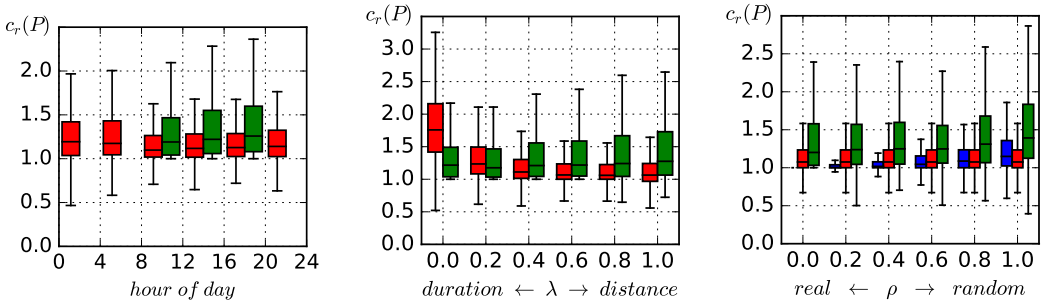
6.2 Cost Weighting

Until now, we weighted all our cost measures equally. Because walking speeds and user preferences may differ, our cost weighting is not universal. To investigate the influence of the weighting parameters on the algorithms, we ran our experiments with a range of different weightings. For each cost weighting $\lambda \in [0, 1]$, we simulated the algorithms with the same starting points. The new weighted cost is defined as $c(P) = 2\big(\lambda \cdot \mathit{distance} + (1 - \lambda)\mathit{duration}\big)$. It is important to note that the algorithms *did* have access to this cost function.

We see that **Heuristic Search** performs well compared to the other algorithms when distance has a larger weight, because its heuristic relies heavily on the distance. All algorithms perform better when duration is weighted high, since it is faster to drive to segments with high probabilities than walking back from them.

6.3 Impact of Inaccurate Probabilities

We can assume that, despite a high number of parking observations, 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



(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 v_0.

(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 $\rho = 0.0$.

Figure 6 Relative algorithm performance $c_r(P)$ divided by the **Branch and Bound** baseline. The respective baseline is represented by just a line. Boxes from left to right: **Branch and Bound**, **Heuristic Search**, and **G2** by Jossé et al. [5].

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 conducted simulations on various noise distributions, but here 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.

Figure 5c compares the costs of our algorithms on noise levels within the interval of $[0, 1]$. As expected, the performance of all algorithms decreases with a rising noise level. Notably, Branch and Bound turns out to be particularly stable against noise and outperforms the other algorithms on all levels. A higher accuracy of the collected probabilities results in a lower cost. Furthermore, we see that until an assumed noise of 0.7, knowledge of the probabilities means an advantage over the heuristic approaches.

7 Conclusion

We proposed two algorithmic approaches for solving the NP-complete problem of parking search. Our **Branch and Bound** algorithm finds the best route within all routes up to a certain length as measured by the cost. For the scenario where no probability data is available, we proposed our **Heuristic Search** algorithm that on average comes close to the baseline by a factor of 1.3 in cost. This leads to the conclusion that per-street probabilities of parking successes are beneficial but not strictly necessary to compute good parking routes. Also, it might be possible to reduce the effort of collecting probability data, since less accurate values still yield attractive results.

We evaluated our algorithms on real per-street probability data for the Berlin area. We released this dataset along with the paper to allow the community to evaluate further parking search algorithms on real data. However, it is unclear how our heuristics would perform in *other cities*. Building on the presented work, two directions of further research sound promising to us. First, we assumed that the probability of a parking spot stays 0 after visiting

it once. It is natural to assume that the observed information decays over time. However, including recovery adds extra assumptions to the model. A *dataset of per-street recovery functions*, e.g., as modeled in [5], would be very interesting in order to explore the impact of recovery in a realistic model. Second, a limitation of our model is to assume independence between the probabilities of nearby roads. Again, real data on this is not available. We conclude that collecting and releasing additional metrics would open the doors to additional research in the field of parking search.

Acknowledgments

We thank TomTom Development Germany GmbH, An den Treptowers 1, 12435 Berlin and its engineering team including our contact persons 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. We are grateful for the permission to release the dataset used in this paper to the public.

References

1 A. Agresti and B. A. Coull. Approximate is better than "exact" for interval estimation of binomial proportions. *The American Statistician*, 52:119–126, 1998.
2 C. R. Cassady and J. E. Kobza. A probabilistic approach to evaluate strategies for selecting a parking space. *Transportation Science*, 32:30–42, 1998.
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*, 25:136–149, 2014.
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*, pp. 347–358, 2010.
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*, pp. 445–456, 2015.
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*, pp. 153–170, 2009.
7 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*, pp. 276–293, 2007.
8 D. Shoup. *The High Cost of Free Parking.* APA Planners Press, 2005.