Probabilistic Resource Route Queries with Reappearance

Gregor Jossé, Klaus Arthur Schmid, Matthias Schubert
Institute for Informatics, Ludwig-Maximilians-University Munich
Oettingenstr. 67, 80538 Munich, Germany
{josse,schmid,schubert}@dbs.ifi.lmu.de

ABSTRACT

In many routing applications, it is unclear whether driving to a certain destination yields the wanted success. For example, consider driving to an appointment and looking for a parking spot. If there are generally few parking spots in the area or if occupancy of spots is currently high, the search may not be successful. In this case, the search is continued, possibly into a different area, where chances of success are higher. We generalize this problem and introduce a probabilistic formalization to model the availability of resources at certain locations. Our probabilistic model considers short term observations (e.g., vacant parking spots) as well as long term observations (e.g., average occupancy time) to adapt to the level of information currently available. In contrast to previous models, we allow resources to reappear after a probabilistically modeled amount of time (e.g., a car leaves a spot). Based on this model, we propose the so-called probabilistic resource route query with reappearance. In order to compute feasible solutions to this query in interactive time, we propose two greedy approaches. Furthermore, we examine backtracking for computing exact solutions and extend the proposed method into a significantly more efficient branch and bound algorithm. In our experiments, we investigate two realistic applications, examine the benefit of our model, and compare algorithmic solutions w.r.t. result quality and computational efficiency.

1. INTRODUCTION

With increasing gas prices, escalating greenhouse gas emissions and heavy traffic congestion in metropolitan areas, optimizing traffic is of great ecological and social importance. While the basic routing task of finding a path from start to target is a well-explored research area, there are other routing tasks common in everyday life which have drawn less attention so far. An example are trip planning queries (TPQ) which are specified by a start location, a target location and a number of resource types which have to be visited along the trip. For instance, the user might provide the resource types “ATM”, “gas station”, and “department store”. The result of such a query is the shortest path from start to target visiting exactly one instance of each resource type. There are several variants to this kind of problem which will be reviewed in Section 2.

A problem relevant to everyday life is guiding a user through a road network to enable them to find a resource for which the availability at certain locations is uncertain. There are several examples for this task. For instance, consider parking spots: although it might be known that certain streets allow parking, it is generally not known whether there will be any vacant spots upon arrival. Another example are drivers of electric cars looking for public charging stations. In some developing countries hospitals with emergency capacities are scarce. Thus, ambulances often visit more than one hospital before being able to hospitalize their patient. In all of these cases, it holds that if the resource is not available upon arrival, the search must be continued to other resource locations until an available resource is found. Hence, guiding a user to the closest resource does not yield a satisfactory solution. Instead, in order to yield a sufficiently large probability of success, a route visiting several resource locations is required. A general problem of finding such routes is that there are two contrary characteristics describing the quality of a route. The first quality measure is the overall likelihood of finding an available resource when following the route. The second quality measure is the cost, e.g., the expected travel time or distance, until the respective resource is found. These two measures are complementary, because continuing the search to another resource location will always increase the success probability but also – without exception – the cost. Thus, it is not possible to minimize the cost while maximizing the probability of success.

In order to compute the success probability of a route, it is necessary to employ information about the availability of resources at all known resource locations. In this paper, we assume a system which collects observations on the availability (and conversely the consumption) of resources over time. These so-called long term observations are then used to compute probability distributions modeling general resource availability. Furthermore, the system provides current information about resource status at query time. We refer to this kind of information as short term observations. For example, long term observations correspond to the average time a parking spot remains vacant or occupied. Short term observations, on the other hand, provide information about currently vacant spots. Depending on the scenario the amount of short term observations might be limited, e.g., only a limited number of parking spots are equipped with occupancy sensors while the rest is detected and reported by other roaming cars. At first glance, short term observations might seem sufficient for a successful search. However, knowing that a resource is available at the moment, does not mean that it still will be upon arrival. Thus, as time progresses, the influence of short term observations on the probability of finding a resource decays. Long term observations address this problem in three ways. First, if there is no short term observation available for a resource, the expected vacancy time of a resource can compen-

sate for the lack of current information. Furthermore, long term observations can be used to predict the probability that a currently available resource will still be vacant upon arrival. And, finally, long term observations can serve as an estimate for the expected occupancy time, i.e., the expected time until a consumed resource becomes available again. For example, in the parking spot scenario this corresponds to the expected parking duration.

In this paper, we present a statistical model describing a road network comprising resource locations of a specific resource type. Our model incorporates both types of information described above, i.e., long term as well as short term observations. From a formal point of view, our model describes each resource location as a continuous-time Markov chain with two states, `available` and `consumed`. Based on this model, we introduce the following query: For a given query location, compute a route for which the probability of finding an available resource exceeds a given probability threshold (e.g., 90 %) while minimizing the (expected) cost, e.g., travel time or distance. A route may be extended infinitely, and each extension adds to the success probability but also to its cost. Thus, in order to optimize one measure, we have to bound the other. More precisely, the best route may be the one with the least cost among all routes exceeding the probability threshold. Or, converesely, the best route may be the one with the highest success probability among all routes not exceeding a cost threshold.

Since our model allows for reapparance of resources, the search space of possible solutions is unlimited. However, in many applications the time frame of finding a suitable answer is rather small as users only tolerate limited answering time. Therefore, we investigate two greedy search heuristics which promise admissible results in efficient time. To allow the computation of optimal results, we examine a recursive backtracking approach to avoid exhaustive search. Furthermore, we propose a lower bound for the remaining increase in cost used in a highly efficient branch and bound solution providing optimal results. We evaluate our approach within real world road networks on the application of finding parking spots and on the application of finding charging stations for electric vehicles. To conclude, the contributions of this paper are as follows:

- A novel probabilistc model describing the availability of resources in road networks. We model resources as continuous-time Markov chains which are parametrized by long term as well as short term observations and allow to model the reapparance of previously consumed resources.
- A new type of query, the Probabilistic Resource Route Query with Reappearance (PRRQR).
- An approximate solution to the PRRQR employing two different search heuristics, as well as optimal solutions based on backtracking and branch and bound.

The rest of the paper is organized as follows: Section 2 summarizes related work about similar types of queries. In Section 3, our probabilistic model is described, followed by the formal definition of the PRRQR. Section 5 describes the heuristics, bounds and query algorithms introduced to process PRRQRs. The results of our experimental evaluation are presented in Section 6. The paper is concluded with a summary and an outlook for future work in Section 7.

2. RELATED WORK

In this section, we survey existing work on similar tasks. First, we will give a review of basic query types related to the one introduced in this paper. Then, we address works which model the existence of resources in a probabilistic way. All of the following query types have the same meta-task, namely, guiding a user to a certain resource. In all of these scenarios, a database which stores the resources and their respective locations is assumed. Although this task may also be carried out in Euclidean space, we restrict ourselves to road networks, as most of the applications are traffic-related.

The simplest type of query guiding a user to the next available resource is the nearest neighbor query (NNQ). In this setting, the user specifies a location – typically his current location – as well as some type of resource. The result of the NNQ is the optimal path (fastest, shortest, etc.) to the closest location providing the resource. As NNQ are a well-explored research area, we will not go into detail on their solutions. An extension of this query are trip planning queries (TPQ) [2] (also referred to as route planning queries [1] or route search queries [10]). In this problem setting, the user specifies a selection of different resources, e.g., “ATM”, “restaurant”, “florist”, “cinema”. Additionally to his start location the user may specify a target location. The result of a TPQ is the optimal path from start to target visiting at least one instance of each resource. Computing TPQs is NP-complete because in the case that each specified resource type occurs exactly once the TPQ degenerates to the Traveling Salesman Problem (TSP).

In another variation of the problem, the order in which resources are visited may be constrained, as described in [9] or [7]. For instance, if planning a date, the order of resources might be restricted by the constraints that the ATM has to be visited first and the florist should be visited before the restaurant and the cinema. However, since the task usually maintains an NP-hard subproblem, solutions to any of these problems typically employ heuristics ([1], [9]).

In the settings presented so far, the existence of a resource at its location is considered to be guaranteed. In many real world applications, however, a resource will only be available with a certain probability. If no table or seats have been reserved, not all locations of type “restaurant” or “cinema” might have the resource available. The same holds for the resource type “florist” if looking for specific flowers. In all of these cases, the requested resource is available with a certain probability (and consumed with the converse probability), i.e., prior to arrival it is not known with certainty whether the resource is available or consumed. At first glance, these kinds of uncertainty may seem congruent. However, there are significant differences which require specific modeling. We distinguish the following types of uncertainty.

Assume a surfer is looking for good waves and is considering different beaches. At every beach, the waves might be sufficient with a certain probability. If available, the resource “waves” cannot be consumed by the presence of other surfers. Thus, we refer to the probability of finding waves as *static (resource) uncertainty*. This uncertainty is independent from the time of arrival and the presence of competitors.

Now, consider a cinema where seats are a limited resource. If no seats have been reserved, the probability of finding seats for a particular show decays as screening time approaches. Since in this case a limited quantity of the resource is consumed over time, we refer to this type of uncertainty as *time-decaying (resource) uncertainty*. Contrastingly, while all tables at a certain restaurant might be occupied now, there might be one available later the same evening. In this case, the quantity of the resource might decay (or more generally: change) over time but regenerate at a later point in time. This is a significant difference to the other scenarios where revisiting resources does not yield any benefit. However, in this scenario, although the resource might have been consumed upon arrival, it might make sense to revisit after an adequate waiting time.

We refer to this kind of uncertainty as *time-dependent (resource) uncertainty with reappearance*.

To the best of our knowledge, there is no previous work supporting short term observations or taking time-dependent uncertainty with reappearance into account. Therefore, we shall now review works which incorporate static as well as time-decaying resource uncertainty. While we provide an abstract problem definition (cf. Section 3) and applications based upon this definition, some works focus on their application and adapt their problem definition to the respective use case. Nevertheless, these works can – with some restrictions – in many cases be extended to incorporate general resources.

The authors of [10], for example, compute paths which guide the user along certain resources. In their follow-up work, [7], this problem is extended by ordering constraints (as in some of the examples above). Both papers present algorithms based on greedy search heuristics as well as on heuristics which minimize the expected distance until search success. In both papers, resources may have assigned success probabilities. However, these probabilities reflect static uncertainty, i.e., non-time-dependent and non-reappearing. The same holds for [4] and its follow-up work [8] where probabilistic k-route queries are introduced and examined. Here, the authors introduce a confidence value which corresponds to the existence probability of a resource at each location, also reflecting static uncertainty. Employing different heuristics, the presented algorithms find approximate solutions by clustering resource locations that maximize the expected success.

In contrast, the authors of [3] take time-dependency into account and assume a linear decay for the vacancy of parking spots. Although this model is aimed at a specific application, it may be generalized to abstract resources. Note, however, that this model does not allow reappearance of resources which is a significant shortcoming, especially in this application. This is because a typical strategy looking for a parking spot is roaming the target area until someone vacates a spot, i.e., a resource reappears. The authors propose two approaches to maximizing the probability of finding a parking spot. The first approach finds the optimal result, however, this is done employing full enumeration on the time-varying TSP. Due to the brute force nature of this algorithm, query processing quickly becomes infeasible with increasing number of resource locations. The second approach is an algorithm that clusters resource locations before solving a TSP on the clusters. Subsequently, the optimal solution within each cluster is searched. Based on a heuristic, this algorithm yields an approximation of the optimal result, while providing a considerable speed-up.

There are various methods, focusing on the application of taxi pickups as well as ridesharing, such as [12], [14],[11] and [5]. In [12], the task is equivalent to solving a classic TSP, i.e., ordering different but fixed pickup locations such that the total distance is minimized. The authors rely on a genetic algorithm approach to solve this problem. In [14],[5] and [11] the task is more complicated. Here, the task is first of all to assign cabs to a set of currently available customers. After this assignment is done, a route for picking up and dropping off the customers has to be found. Thus, only the last part of the query is related to our work. Furthermore, none of the works considers uncertainty w.r.t. customer availability.

3. PROBLEM SETTING

In this section, we formalize our problem setting. First, we define the graph which represents the underlying road network. Then, we introduce the probabilistic model which describes resource availability and consumption.

3.1 Road Network Graph

For a given road network, we let $G = (V, E)$ denote the corresponding graph, i.e., the vertices (or nodes) $v \in V$ correspond to crossings, dead ends, etc., and the edges $e \in E \subseteq V \times V$ represent directed road segments connecting the vertices. We refer to this graph as *Road Network Graph*. Furthermore, let $c : E \to \mathbb{R}_0^+$ denote the function which maps every edge onto its respective cost, e.g., travel time or distance. If the employed cost function is not travel time, we additionally assume the travel time to be known and given by a function $t : E \to \mathbb{R}_0^+$ (as resource availability is dependent on the time of arrival). By *route*, we mean a consecutive set of edges (possibly with cycles), i.e., $r = (e_1, \ldots, e_n)$ where all e_i are taken from the corresponding set of edges and for all $1 \leq j \leq n-1 : e_i = (u, v) \Rightarrow e_{i+1} = (v, w)$. The cost of a route $r = (e_1, \ldots, e_n)$ is defined as the accumulated cost of its edges, i.e., $c(r) = \sum_{i=1}^{n} c(e_i)$. By *path*, we mean a cycle-free route.

3.2 Probabilistic Model

In the following, we introduce our probabilistic model. As mentioned before, at every resource location the respective resource may either be `available` or `consumed`. However, prior to arrival at the location, it is not known which of the two is the case. Our probabilistic model has to be able to reflect all three kinds of uncertainty: static, time-decaying, and time-dependent uncertainty with reappearance. While the former two kinds of uncertainty are rather straightforward, the latter requires more work and a novel approach.

The static uncertainty of a resource is easily expressed by a random variable X which takes the values 0 or 1, representing the states `available` and `consumed`, respectively, where the probabilities of X are $\mathbb{P}(X = 0) = p$ and $\mathbb{P}(X = 1) = 1 - p$ for some $p \in [0, 1]$. For illustration, recall the example of a surfer looking for waves at a beach. Independent of the time of their arrival there will be waves with probability p.

In the case of time-decaying uncertainty, we propose modeling the probability that a resource X is available at time $t > 0$ as $e^{-\lambda t}$ for some $\lambda > 0$. Consequently, at time $t = 0$, the resource is available with probability 1, but it decreases as time progresses and asymptotically approaches 0. Or, in other words, the probability that X is consumed is the cumulative distribution function of an λ-exponentially distributed random variable. This coincides with the intuition of modeling waiting times as exponentially distributed random processes. For illustration, recall the example of buying tickets to the movies, where the probability of available seats is 1 at $t = 0$ but decreases as screening time approaches.

Now, let us turn to the case of time-dependent uncertainty with reappearance which is the core of this work, as it is the only concept that can model the use cases of our experiments (parking spots, charging stations). As before, resource availability has two states, but now, there may occur multiple state transitions at arbitrary points in time. Therefore, we propose modeling each resource as a stochastic process. The most common type of stochastic processes are Markov chains which model the transition probabilities within a system with a given number of states. When a system transitions from one state into another, the future state is only dependent on the present state. This property is central to Markov chains and referred to as memorylessness or Markov property. Markov chains can either assume discrete or – as in our case – continuous time. In a discrete model, there exist equal time steps, and for each step, the probability of transitioning into another state can be computed. In a continuous-time model, the sojourn time in each state, i.e., the time until the next state transition, is perceived as a random vari-

able itself. The notion of memorylessness extends naturally to the case of continuous time.

Thus, we model time-dependent resource availability with reappearance at each resource location as a continuous-time Markov chain (CTMC). More precisely, each resource location r^i is now represented by a family of random variables $\{X_t^i, t \geq 0\}$ with values in the state set $\{0, 1\}$. Note that there exists a one-to-one relationship between each resource location, and its resource modeled by the respective CTMC. Thus, we denote the CTMC of each resource location r^i by X^i and denote the set of all CTMCs by $\mathcal{X}$, where $|\mathcal{X}| = |\mathcal{R}|$. We may use the term *resource* for the geolocation associated with a resource as well as for the corresponding CTMC. When not clear from the context, we will state explicitly which of the two is referred to. Also, we assume the resources, more specifically their CTMCs, to be mutually independent. The independence assumption keeps the model general and applicable even if the available observations are limited.

Besides its state space, a CTMC is (under the reasonable assumption of time-homogeneity) defined by the family of transition matrices $\{P_t, t \geq 0\}$ and the (infinitesimal) generator matrix Q. Using the Kolmogorow equations, each may be computed from the other by solving the first order differential equation $P'(t) = P(t)Q$. For more mathematical details, we refer the reader to [13]. We omit the explicit calculations here and restrict ourselves to explaining the connection between $P(t), Q$ and the states of a resource.

In our case, Q is a 2×2-matrix. Its diagonal entries reflect the parameters of the random variables modeling the sojourn time of each state while the non-diagonal entries reflect the rate of transition into another state. Q has the following form:

$$Q = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

The family of transition matrices $P(t)$ for $t \geq 0$ is defined as follows:

$$P(t) = \begin{pmatrix} \frac{\mu}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu}e^{-(\lambda+\mu)t} & \frac{\lambda}{\lambda+\mu} - \frac{\lambda}{\lambda+\mu}e^{-(\lambda+\mu)t} \\ \frac{\mu}{\lambda+\mu} - \frac{\mu}{\lambda+\mu}e^{-(\lambda+\mu)t} & \frac{\lambda}{\lambda+\mu} + \frac{\mu}{\lambda+\mu}e^{-(\lambda+\mu)t} \end{pmatrix} \quad (1)$$

For each resource, the sojourn times of its states `available` and `consumed` are modeled as exponentially distributed random variables with parameters λ and μ, respectively. This, again, coincides with the convention of modeling waiting times as exponentially distributed. The expected value of an exponential distributed random variable $\exp(\phi)$ is $1/\phi$. Thus, the expected sojourn time in the state of availability is $1/\lambda$, and the expected sojourn time in the state of consumption is $1/\mu$. One application on which we evaluate our model in Section 6 is finding vacant parking spots. In this use case, $1/\lambda$ would be the expected vacancy time, analoguously, $1/\mu$ would be the time until an occupied spot becomes vacant again. Of course, these parameters may be different for each resource location if enough observations for an individual estimation are available. Therefore, it is possible for each resource to have distinctly parametrized Q and $P(t)$.

Let us shortly explain how the assumed observations are used for parameter estimation. As mentioned before, our model allows to incorporate short term as well as long term observations. The latter are used for parameter estimation in the following way: Consider a resource X, which has an unknown expected sojourn time in the state `available` but is assumed to be exponentially distributed with parameter λ. Given a number of observations $x = (x_1, \ldots, x_r)$, i.e., exemplary measurements of the time span during which X stays available, we can easily estimate λ using the maximum likelihood estimator. The according likelihood function is given by:

$$L(\lambda) = \prod_{i=1}^{r} \lambda \exp(\lambda x_i) = \lambda^r \exp(-\lambda r \bar{x})$$

where $\bar{x} = 1/r \sum x_i$ denotes the mean of all measurements. Differentiating the logarithmized likelihood function yields the maximum likelihood estimator $\hat{\lambda} = 1/\bar{x}$, which is simply the inverse of the mean value. The parameter μ may of course be estimated analoguously.

Now, let us review some properties of the transition matrices $\{P(t), t \geq 0\}$ central to the model. Given a resource X and its sojourn time parameters λ and μ, we can compute $P(t)$ as in Equation 1. If we also have an initial probability distribution based on short term observations of the states of X at time $t_0 = 0$ (denoted as π_0), we can compute the according probability distribution after an arbitrary point in time $t \geq 0$ (denoted as π_t) as follows:

$$\pi_t = \pi_0 P(t)$$

Note that $P(0) = I$ is the identity matrix (cf. Equation 1) which means that if no time has passed, the probability distribution of t_0 is still active. For example, if there is an observation at t_0 of X being in state `consumed`, then $\pi_0 = (0, 1)$. The consumption of resource X is certain – but only at this particular point in time. As time progresses, it becomes more likely that the state changes. Therefore, the original probability distribution π_0 (given by the observation) changes. Note that this reflects the notion of reappearance of previously consumed resources. This is expressed in the (exponentially) decaying influence of the second summands in every entry of $P(t)$ (cf. Equation 1). Eventually, the original observation becomes obsolete. This can be seen from the convergence of $P(t)$ as $t \to \infty$:

$$\lim_{t \to \infty} P(t) = \begin{pmatrix} \frac{\mu}{\lambda+\mu} & \frac{\lambda}{\lambda+\mu} \\ \frac{\mu}{\lambda+\mu} & \frac{\lambda}{\lambda+\mu} \end{pmatrix}$$

Asymptotically both rows of $P(t)$ are equal. This implies the initial observation has no more influence on the probability distribution of X as $t \to \infty$. For example, whether the initial observation was `consumed`, i.e., $\pi_0 = (0, 1)$, or `available`, i.e., $\pi_0 = (1, 0)$ is without significance. In any case, $\lim_{t \to \infty} \pi_t$ is the so-called stationary distribution as introduced below.

In our case, a resource X is a finite-state Markov chain where all states communicate and thus X has a unique stationary distribution π [13]. By definition: $\pi P(t) = \pi, \forall t \geq 0$. Solving this system of equations, we get:

$$\pi = (\pi_1, \pi_2) = \begin{pmatrix} \frac{\mu}{\lambda+\mu} & \frac{\lambda}{\lambda+\mu} \end{pmatrix}$$

This is equal to the rows of $\lim_{t \to \infty} P(t)$ which supports the intuition of t having no more influence on the probability distribution. For example, if no observation for X is available, the only unbiased assumption is the stationary distribution since it assumes the respective share of both states w.r.t. λ and μ.

Let us use all of the above in an example related to one of our applications: Let X be a CTMC modeling the vacancy of a parking spot at a certain location. We make the following assumptions on the model: If `available` (meaning vacant), we expect X to be occupied within 5 minutes. This means, the sojourn time of state `available` is a $1/5$-exponentially distributed random variable. If X is occupied, we expect the occupant to leave within 20

minutes. Hence, the sojourn time of state `consumed` (meaning occupied) is a $1/20$-exponentially distributed random variable. As mentioned before, $P(0) = I$. Let us investigate how $P(t)$ changes as time (non-infinitely) progresses. For instance, after 1 and after 3.5 minutes:

$$P(1) \approx \begin{pmatrix} 0.8 & 0.2 \\ 0.05 & 0.95 \end{pmatrix} \quad P(3.5) \approx \begin{pmatrix} 0.52 & 0.48 \\ 0.12 & 0.88 \end{pmatrix}$$

Assume that at $t_0 = 0$ X has been observed as `available`, i.e., $\mathbb{P}(X_0) = (1, 0)$. Then at $t = 1$ the spot will still be `available` with probability 0.8. After 3.5 minutes this probability will have decreased to 0.52. Now, consider a different scenario where at $t = 0$ X has been observed as `consumed`, then after 1 minute it is `available` with probability 0.05. After 3.5 minutes this probability will have increased to 0.12.

To conclude, we now have a probabilistic model on our hands which is capable of describing all three kinds of resource uncertainty introduced in Section 2. The core contribution of this model is its ability to reflect resource reappearance. Also, it allows efficient resource-specific parametrization by incorporating long term and short term observations.

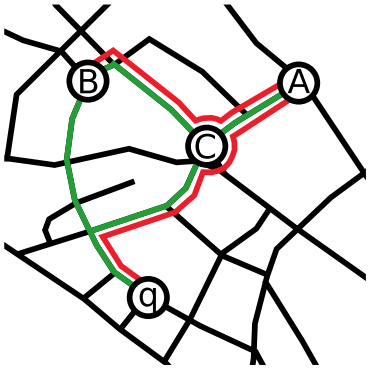
4. QUERY DEFINITION AND RESULT SET

Now that we have defined the probabilistic model, we turn to our query and its result. Both are best described using an alternative graph, referred to as *resource graph* $\hat{G}$. Therefore, in this section, we will first define the resource graph. Subsequently, we introduce two measures which are then used to define the Probabilistic Resource Route Query with Reappearance (PRRQR) and its result.

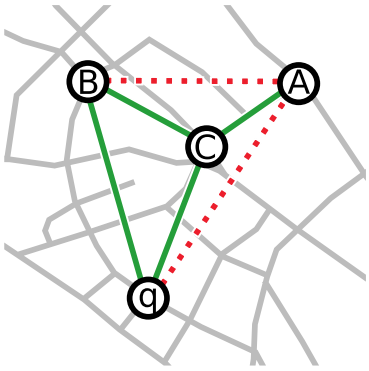
4.1 Resource Graph

We assume that for a road network graph and a query node q a set of suitable resources $\mathcal{X}(q) = \{X^1, \ldots, X^N\}$ is given. In the majority of applications, only a reasonable subset of resources might qualify. For example, parking spots should be within walking distance of the driver's destination. In this case, the query node would be the driver's position when he reaches the vicinity of his destination. An according range query to a database would then retrieve suitable parking opportunities on which a PRRQR would be executed. For every resource X^i, we assume distribution parameters λ^i, μ^i and an initial distribution π_0^i as given.

The set of vertices of the resource graph is defined as $\hat{V} := \{q\} \cup \mathcal{X}$, where $q \in V$ denotes the query node. The edges of the resource graph, $\hat{E}$, represent cost-optimal paths between resource locations in the underlying road network. Thus, each route in the resource graph can be expanded into a corresponding route in the transportation network (cf. Figure 1(a)). Let us note that even in cases where cost does not refer to the travel time, we will also compute the travel time of cost-optimal path because it is needed to for estimating the success probability. Although it is possible to compute the cost-optimal path between each pair of resource locations, we will require $\hat{E}$ to contain only a minimal set of edges by removing transitive connections. A transitive connection is a path within the road network that contains at least one intermediate resource location. For example, if along the cost-optimal path from X^A to X^B resource location X^C is encountered, then X^A and X^B would not be connected in the resource graph (cf. Figure 1(b)). However, $\hat{G}$ would contain edges connecting X^A and X^C as well as X^C and X^B. We explicitly exclude transitive connections from the resource graph for two reasons. First, transitive links do not allow computing the success probability correctly because the intermediate resource locations are not considered. Second, the existence of transitive connections leads to the inefficient traversal of identical subpaths.



(a) Optimal direct paths between resources are marked green, fastest paths with intermediate resources are marked red.



(b) Edges of the resource graph (marked continuously green) and excluded transitive connections (marked dashed red).

Figure 1: Illustration of a query node q and resources A, B, C in a road network graph (a) and the respective resource graph (b).

We also compute the cost-optimal paths from the query node q to all resource locations. But as there is no gain in returning to the query node, paths ending at q are excluded from the computation. Algorithm 1 describes the computation of $\hat{E}$ which is illustrated in Figures 1. Note that in order to compute $\hat{E}$, we need to compute all cost-optimal paths within the road network graph and then prune the transitive connections. It is not possible to avoid the computation of transitive connections directly.

The cost function of the road network graph G naturally extends to $\hat{G}$. Since every edge in $\hat{G}$ corresponds to an cost-optimal path in G, the cost function $\hat{c} : \hat{E} \to \mathbb{R}_0^+$ maps an edge of $\hat{E}$ to the accumulated costs of the respective path in G. Analogously, $\hat{t} : \hat{E} \to \mathbb{R}_0^+$ maps an edge of $\hat{E}$ to the accumulated time of the respective cost-optimal path in G.

Combining the above, we define the resource graph as $\hat{G} = \hat{G}_{(q,\mathcal{X})} := (\hat{V}, \hat{E}, \hat{c}, \hat{t})$. Note that $\hat{G}$ holds all the query-relevant information since it contains the query node q as well as the resource locations $\mathcal{X}(q)$. Therefore, speaking of a query setting, we mean q and $\mathcal{X}(q)$ as well as the according resource graph $\hat{G}$.

Algorithm 1: Computation of $\hat{E}$

Input: Query setting $\mathcal{X}(q), q$
Output: Edges $\hat{E}$ of resource graph $\hat{G}$

```
begin
    Ê ← ∅
    foreach X ∈ 𝒳(q) do
        Compute fastest path p from q to X
        if no intermediate resource on p then
            Ê ← Ê ∪ p
        end
        Compute multi-target Dijkstra from X to all
        X' ∈ 𝒳 \ X in G
        foreach fastest path p from X to X' do
            if no intermediate resources on p then
                Ê ← Ê ∪ p
            end
        end
    end
end
```

4.2 Resource Routes and PRRQR

Relying on $\hat{G}$, we may now define the possible solutions to our query. For a given query setting $q, \mathcal{X}(q)$ and the according resource graph $\hat{G}$, a *resource route* is a route r in $\hat{G}$, starting at query node $q = X^0$. Note that by construction of the resource graph a resource route only follows optimal paths between resources. We describe a resource route as a set of edges (in $\hat{G}$), i.e., $r = (e_{r_1}, \ldots, e_{r_n})$, where $e_{r_i} \in \hat{E}$ for all $1 \leq i \leq n$. Since in our context the order of resources is of particular interest, we introduce a specific notation for it. Recall that every edge in $\hat{E}$ connects two resources unless it starts at the query node. Hence, along a resource route r with n edges we encounter n resources. Note that these resources are not necessarily distinct, as r may contain cycles. Let $X_r = (X^{r_1}, \ldots, X^{r_n})$ denote the n resources along r. To introduce a measure for the success probability of a complete route, we start by defining the success probability of a resource route.

DEFINITION 1. *For a given resource route r along resources $(X^{r_1}, \ldots, X^{r_n})$, let t_i denote the* time of arrival *at X^{r_i}, i.e., $t_0 < t_1 < \cdots < t_n$, and let c_i denote the accumulated cost up to resource X^{r_i}. Furthermore, let $\{P(X^{r_i}, t), t \geq 0\}$ denote the transition matrices of X^{r_i} (dependent on the parameters of X^{r_i}) and let $\pi_0(X^{r_i})$ denote the initial distribution of the states of X^{r_i} (dependent on the availability of short term observations regarding X^{r_i}). Then the probability distribution of X^{r_i} at t_i is defined as:*

$$\big(\mathbb{P}(X^{r_i}_{t_i} = 0), \mathbb{P}(X^{r_i}_{t_i} = 1)\big) := \pi_{t_i}(X^{r_i}) = \pi_0(X^{r_i})P(t_i)$$

Hence, the success probability *of X^{r_i} at arrival time t_i is the probability that resource X^{r_i} is in state* `available` *at time t_i, i.e., $\mathbb{P}(X^{r_i}_{t_i} = 0)$.*

Note that in accordance with the probabilistic model as presented in Section 3, we denote state `available` of a resource X by $X = 0$ and the state `consumed` by $X = 1$. Based on this definition, we are able to define the success probability for a complete resource route:

DEFINITION 2. *Let $q, \mathcal{X}(q), \hat{G}$ be a query setting and r be a resource route in $\hat{G}$. The* Success Probability *of r, denoted by $\mathbb{P}_S(r)$, is defined as the probability of the complementary event of not finding any available resource along r:*

$$\mathbb{P}_S(r) := 1 - \left(\prod_{i=1}^{n} \mathbb{P}(X^{r_i}_{t_i} = 1)\right)$$

Given the success probability, we now need to find a second measure which measures the effort of finding an available resource, i.e., the expected cost of a resource route:

DEFINITION 3. *Let $q, \mathcal{X}(q), \hat{G}$ be a query setting and r be a resource route in $\hat{G}$. The* Expected Cost *of r, denoted by $\mathbb{E}_c(r)$, is defined as:*

$$\mathbb{E}_c(r) := \sum_{i=1}^{n} \left(c_i \cdot \mathbb{P}(X^{r_i}_{t_i} = 0) \cdot \prod_{j=1}^{i-1} \mathbb{P}(X^{r_j}_{t_j} = 1)\right)$$

Relying on both measures, we can now define the *Probabilistic Resource Route Query with Reappearance (PRRQR)*:

DEFINITION 4. *Let q be a query node, $\mathcal{X}(q)$ the set of corresponding resources, $\hat{G}$ the according resource graph. Furthermore, let $0 \ll \rho < 1$ denote a probability threshold. The result of a PRRQR with threshold ρ, denoted by PRRQR(ρ), is the resource route in $\hat{G}$ with minimal expected cost among all resource routes which exceed the probability threshold ρ.*

$$\textit{PRRQR}(\rho) = \arg\min\{\mathbb{E}_c(r) \mid \mathbb{P}_S(r) \geq \rho\}$$

Note that there exists a straightforward variation of the PRRQR. Instead of thresholding the success probability, one could bound the maximal cost. Given a cost threshold $\tau > 0$, the query result is the resource route maximizing the success probability while not exceeding the cost bound τ. In this case, it is usually more reasonable to employ τ as a strict bound on the maximal cost instead as a bounding for the expected cost. For example, consider driving an electric vehicle with a limited remaining range. Bounding the expected distance misses the point, only bounding the actual driven distance (while maxmimizing the success probability) will provide a suitable route. This variation of the query is also examined our experiments. However, in the following, we focus on the first (and more sophisticated) case to keep the description compact.

5. QUERY PROCESSING

In this section, we present algorithms for processing PRRQRs. First, we propose two heuristics which are employed in a greedy search that computes approximations of the optimal results. Afterwards, we will present two methods computing optimal solutions, relying on backtracking as well as on branch and bound. In the following, let $0 < \rho < 1$ be a probability threshold, and let $\hat{G}$ be the resource graph according to a given query setting $q, \mathcal{X}(q)$.

Let us note some aspects central to the PRRQR before going into the details of the algorithms. Given $\hat{G}$ and the query position q, then all resource routes start at q by definition. Hence, the set of possible solutions may be conceived as a search tree rooted at q where each branch is a sequence of resource locations. For a probability-constrained PRRQR with $\rho = 1$, i.e., a PRRQR requiring certainty of finding a resource, this search tree is infinite. The effect is caused by the time-dependent decay inherent in our model. Thus, a certain observation of availability of a particular resource will no longer be certain at the time of arrival. As a result, the success probability of a resource route can only asymptotically converge against 1. Even when $\rho < 1$, the search space is generally very large. This is because considering resource reappearance adds considerably to the

complexity of the task. Similar to the Traveling Salesman Problem (TSP), there is no local optimality w.r.t. the resource subsequences that can be exploited. Hence, it is not possible to tell whether a resource route r can be extended into an optimal solution based on its current $\mathbb{P}_{\mathrm{S}}(r)$ and $\mathbb{E}_{\mathrm{c}}(r)$. Furthermore, it is easy to see that all permutations of the set of resources can be found in the search tree. Thus, from a theoretic point of view the problem is NP-complete. Only by setting the threshold $\rho < 1$, the search space becomes finite.

One precomputational step which all algorithms have in common is the computation of the resource graph $\hat{G}$ and its edges which constitute cost-optimal paths between resource locations in the underlying road network graph. The pseudocode for this operation is given in Algorithm 1. The set of edges $\hat{E}$ is realized as an adjacency matrix A of dimension $N + 1 \times N$, where $|\mathcal{X}(q)| = N$ is the number of suitable resources of the respective query setting. For notational reasons, we denote the query node q by X^0. The entries $a_{ij}, 0 \leq i \leq N, 1 \leq j \leq N$ of A are defined as:

$$a_{ij} = \begin{cases} c(p(X^i, X^j)) & \nexists X^k \in p(X^i, X^j), i, j, k \text{ pairw. inequal} \\ \infty & else \end{cases}$$

where $p(X^i, X^j)$ denotes the cost-optimal path from X^i to X^j within the underlying road network graph. A holds the cost of all cost-optimal paths, both pairwise between resources as well as from q to any resource, if no intermediate resources are located along this path. A, however, does not hold any information about paths from any resource to q, because the query node is not a resource and thus does not yield any gain toward the query goal. In case cost does not refer to travel time, we also compute the travel times of the cost-optimal paths. Recall the example of a query setting and its resource graph, as illustrated in Figure 1.

The according adjacency matrix of this scenario is given by:

	A	B	C
q	∞	$t(p(q, B))$	$t(p(q, C))$
A	∞	∞	$t(p(A, C))$
B	∞	∞	$t(p(B, C))$
C	$t(p(C, A))$	$t(p(C, B))$	∞

As mentioned before, depending on the application, the subset of suitable resource locations may be query-dependent. However, as the total set of resource locations is known prior to the query, it is possible to precompute the above adjacency matrix. If at query time a selection of suitable resources is required, the non-relevant rows and columns may simply be ignored. In the following, we consider an appropriate adjacency matrix as given. This assumption is not to the disadvantage of any of the proposed algorithms, as they all rely on the resource graph $\hat{G}$ and its edges modeled by the adjancency matrix.

5.1 Heuristic Solutions

In order to cope with the complexity of the PRRQR, we propose two search heuristics employed in greedy algorithms. Both heuristics aim at exceeding the given probability threshold by extending a (partial) resource route r by the "best" next resource location. The first heuristic greedily chooses the resource location which yields the best success probability upon arrival, while the second heuristic greedily chooses the resource location which yields the best tradeoff between success probability upon arrival and cost to reach the location from the present one. Formally, we propose to evaluate a possible extension of resource route r along resources $(X^{r_1}, \ldots, X^{r_{i-1}})$ by one of the resource locations $\{X^1, \ldots, X^N\}$ according to the following heuristics:

(1) Extend r by X^{r_i} such that

$$X^{r_i} = \arg\max\{\mathbb{P}(X_{t_j}^{r_j} = 0) \mid 1 \leq j \leq N\}$$

(2) Extend r by X^{r_i} such that

$$X^{r_i} = \arg\max\{\mathbb{P}(X_{t_j}^{r_j} = 0)/\hat{c}(e_{r_j}) \mid 1 \leq j \leq N\}$$

where $\hat{c}(e_{r_j})$ denotes the cost for traveling from resource location $X^{r_{j-1}}$ to X^{r_j} along the an cost-optimal path.

In conclusion, our greedy approaches **G1** and **G2** proceed as follows: For a given query setting $q, \mathcal{X}(q)$ and a probability threshold $0 < \rho < 1$ all cost-optimal paths from q to all resource locations adjacent to q w.r.t. the adjacency matrix are computed. Then, **G1** and **G2** choose the most promising extension according to the extension strategies (1) and (2), respectively. If the success probability of the obtained resource route does not exceed the probability threshold ρ, the procedure is repeated for all resource locations adjacent to the current one w.r.t. the adjacency matrix. As soon as the success probability exceeds ρ, we have found a viable solution.

5.2 Optimal Results

The greedy approach described above aims at the computation of reasonable resource routes in efficient time. However, in some applications, quality is more important than efficiency. For these cases, we propose two different approaches which guarantee optimal results. We present a backtracking algorithm and a further accelerated branch and bound approach.

5.2.1 Optimal Results through Backtracking

The backtracking approach, denoted by **BT**, starts at query node q and gradually expands resource routes as long as they qualify as result candidates. A resource route disqualifies as a result candidate if it exceeds the expected cost of the currently best resource route. During the expansion, **BT** explores the search tree (rooted at q) in depth-first order. Note that this search tree is generally infinite. Consequently, it is of even greater importance to exclude resource routes from expansion early on in the algorithm. Therefore, we conduct a prior initialization step equal to an execution of the greedy **G2** algorithm. This generates a valid resource route in efficient time, its expected cost may be used as a first bound. We omit the initialization step here (since it is equal to the description of **G2** above). Instead, we only give the recursive procedure as presented in Algorithm 2.

The procedure `expandRecursive` is initially called with a trivial resource route only consisting of query node q and an unspecified resource (which is reset to an adjacent resource during the first run). The expected cost of the result generated by **G2** is held in a global variable M_{c} as an initial upper bound for the expected cost. While traversing the search tree M_{c} will be tightened by finding better solutions. In line 3, candidates which do not qualify as results are excluded, while in line 6 possible result are generated (i.e., resource routes exceeding the probability threshold). The actual search tree traversal is realized recursively in lines 10, 11. If an expansion r' of a resource route r is better than the current best route along this subtree $\hat{r}$ (w.r.t. the expected cost), then $\hat{r}$ is updated to r' and M_{c} is updated to $\mathbb{E}_{\mathrm{c}}(r')$ (lines 13,14). Thus, by sequential traversal of the search tree, **BT** returns the optimal result upon termination. However, due to the exponential number of branches and the technically infinite length of the branches runtime is prone to degenerate. Therefore, we propose another algorithm which computes optimal results in significantly less time.

Algorithm 2: Expansion step of **BT**

```
1  expandRecursive(Resource route r,
2  resource X)
   Data: Upper bound for E_c, M_c
   Output: Optimal resource route r̂
3  begin
4      if E_c(r) > M_c then
5          return ∅
6      end
7      if P_S(r) > ρ then
8          return r
9      end
10     initialize variable holding current best route r̂ = ∅
11     foreach resource X adjacent to last resource of r
       do
12         r' ← expand(r, X)
13         if r̂ = ∅ ∨ E_c(r') < M_c then
14             r̂ = r'
15             M_c ← E_c(r̂)
16         end
17     end
18     return r̂
19 end
```

5.2.2 Optimal Results through Branch and Bound

Like the backtracking algorithm **BT**, this branch and bound approach, denoted by **BB**, relies on an upper bound for the expected cost (M_c) which is tightened as the algorithm progresses. Additionally, **BB** incorporates a forward estimation for the expected cost a route minimally needs to exceed the probability threshold ρ. The forward estimation is a lower bound for the expected cost w.r.t. a resource route and ρ. Consequently, if this lower bound exceeds the upper bound for the expected cost M_c, r can be excluded from further expansion, i.e. the respective subtree can be pruned.

Algorithmically, **BB** is similar to **BT**, except for the mentioned forward estimation. This forward estimation is incorporated into Algorithm 2 as an if$(m_c < M_c)$-statement spanning from line 11 through line 17, where m_c is the output of procedure `forwardEstimation`, as presented in Algorithm 3. Before each possible expansion of a resource route r, `forwardEstimation` is called with r and the probability threshold ρ. In lines 3-7 the parameters are set which are subsequently used to compute the lower bound for the expected travel time. t_{now} is the absolute travel time of input resource route r. t_{min} is the fastest travel time between any two resources. Thus, t_{opt} is the minimal possible arrival time at the next resource w.r.t. the absolute travel time of r. p_{opt} and m_c are initialized with $\mathbb{P}_S(r)$ and $\mathbb{E}_c(r)$, respectively. Both values are updated in the while loop (lines 8-13) until $p_{\text{opt}} \geq \rho$, i.e. until the optimal success probability exceeds the threshold. Now, let us investigate the operations in the while loop. First, p_{max} is defined as the maximal probability among all resources at the minimal possible arrival time t_{opt}. Note that we only allow observations to be incorporated into the model until query time. Therefore, p_{max} is monotonically decreasing in t_{opt}, and it converges against the minimal value of all stationary distributions in state `consumed`. m_c is extended by a new summand reflecting a hypothetical and optimal journey to the resource with maximal probability and minimal cost. Consequently, the success probability bound is updated to the probability of the complementary event of not finding any available resource along this optimal journey. By this strategy, in every iteration of the while loop, a journey to an optimal next resource causing minimal cost is simulated. Thus, the maximal success probability is aggregated while assuming minimal cost. We prove this in the following lemmas. We introduce the following terminology: For a given resource route r we refer to any iteration of the while loop of Algorithm 3 as an *optimal extension*. This coincides with the above described intuition.

Algorithm 3: Forward Estimation of **BB**

```
1  forwardEstimation(Resource route r,
   current E_c bound M_c)
   Output: Upper bound for the success probability
           of any extension of r until it exceeds M_c
2  begin
3      t_now ← arrival time at last resource of r
4      t_min ← min_{e∈Ê} t̂(e)
5      t_opt ← t_now + t_min
6      p_opt ← P_S(r)
7      m_c ← E_c(r)
8      while m_c < M_c do
9          p_max ← max_{X∈X(q)} P(X_{t_opt} = 0)
10         m_c ← m_c + (t_opt · p_max(1 − p_opt))
11         p_opt ← 1 − ((1 − p_opt)(1 − p_max))
12         t_opt ← t_opt + t_min
13     end
14     return p_opt
15 end
```

LEMMA 1. *In any optimal extension the gain of the updated values $m_c' \leftarrow m_c$ and $p_{opt}' \leftarrow p_{opt}$ yield the best possible trade-off between expected cost and success probability. More specifically, let r be a resource route with $\mathbb{E}_c(r) = m_c$, $\mathbb{P}_S(r) = p_{opt}$ and travel time t_{opt}. Then the following statement holds: For any possible extension r' of r to another resource:*

$$\frac{m_c' - m_c}{p_{opt}' - p_{opt}} < \frac{\mathbb{E}_c(r') - \mathbb{E}_c(r)}{\mathbb{P}_S(r') - \mathbb{P}_S(r)}$$

PROOF. *In order to prove this lemma, we need to formulate the success probability of a resource route r differently:*

$$\begin{aligned}\mathbb{P}_S(r) &= 1 - \prod_{i=1}^{n} \mathbb{P}(X_{t_i}^{r_i} = 1) \\ &= \sum_{i=1}^{n} \left(\mathbb{P}(X_{t_i}^{r_i} = 0) \cdot \prod_{j=1}^{i-1} \mathbb{P}(X_{t_j}^{r_j} = 1) \right)\end{aligned}$$

Note that the equality indeed holds. This is because the event of finding at least one available resource can be described by the complementary event of not finding any resource in state available. Equally, it can be described as the union of events that resource X^{r_i} is available but all other resources thus far were consumed. Now, for a given resource route r, let r' denote an arbitrary extension of r by another resource X with respective arrival time t. By the alternative definition of $\mathbb{P}_S$, we have $\mathbb{P}_S(r') = \mathbb{P}_S(r) + t\mathbb{P}(X_t = 0) \cdot (1 - \mathbb{P}_S(r))$. Recall that $\mathbb{E}_c(r) = m_c$ and $\mathbb{P}_S(r) = p_{opt}$.

Now, we show our claim:

$$\frac{m'_c - m_c}{\mathbb{E}_c(r') - m_c} < \frac{p'_{opt} - p_{opt}}{\mathbb{P}_S(r') - p_{opt}}$$

$$\frac{c'_{opt} p'_{opt}(1 - \mathbb{P}_S(r))}{c(r)\mathbb{P}(X_t = 0)(1 - \mathbb{P}_S(r))} < \frac{p'_{opt}(1 - \mathbb{P}_S(r)}{\mathbb{P}(X_t = 0)(1 - \mathbb{P}_S(r)}$$

By definition, $t'_{opt} \leftarrow t_{opt} + t_{min}$, where t_{min} denotes the minimal travel time in $\hat{E}$. Consequently, $t'_{opt} < t$, therefore, the inequality holds which proves the claim. □

LEMMA 2. *Let r be a resource route. The forward estimation of the expected cost as computed by Algorithm 3 is indeed a lower bound.*

PROOF. *This follows from the following properties:*

(i) The number of optimal extensions needed until r exceeds the probability threshold is at most the number of actual extensions needed.

(ii) Every optimal extension of r yields a better trade-off than an actual extension.

(iii) No sequence of actual extensions of r can exceed the probability threshold while yielding a lower expected cost than the sequence of optimal extensions chosen by Algorithm 3.

(i) follows directly from the definition of $p_{opt} \leftarrow 1 - \big((1 - p_{opt})(1 - p_{max})\big)$. In every optimal extension, p_{opt} is increased by the maximally possible value. Therefore, no other sequence of extensions can yield a faster increase. (ii) is the statement of Lemma 1. Finally, (iii) follows from both, (i) and (ii). □

Note that all of the above is easily applied to the case where instead of minimizing the expected cost w.r.t. a probability threshold we maximize the probability w.r.t. an absolute cost bound (as introduced at the end of Section 4.2). For example, consider the backtracking expansion Algorithm 2. Instead of dismissing (storing) a route r if $\mathbb{E}_c(r) > M_c$ ("$<$" holds), in the complementary scenario, a route r is dismissed (stored), if $c(r) > M_c$ ("$<$" holds). Similarly for the forward estimation presented in Algorithm 3. Again, the expected cost $\mathbb{E}_c(r)$ is to be replaced with the absolute cost $c(r)$. While the absolute cost bound is not exceeded, optimal path extensions are simulated, adding maximal success probability to the path. When the cost bound is exceeded and the maximal current best success probability is not surpassed, the search tree can be pruned. If, on the other hand, the success probability is surpassed by the optimal path extension, the path (and its subtree in the search tree) qualifies as a candidate. As for the theoretic arguments, they apply analoguously, therefore we omit an adapted version due to space limitations.

Concludingly, we have presented four algorithms for solving the proposed PRRQR in this section. **G1** and **G2** follow a greedy heuristic to produce approximate results, while **BT** and **BB** produce exact results. **BB** is an extension of **BT** which makes use of a lower bound forward estimation of the expected cost. In the above lemmas, we have shown correctness of the proposed bound.

6. EXPERIMENTAL EVALUATION

We evaluate our model and our algorithms on settings in real world road networks extracted from OpenStreetMap[1] (OSM) using the MARiO framework [6]. All experiments were conducted on a desktop computer equipped with an Intel Core i7-3770 CPU and 32 GB RAM, running Java 1.64 (64-Bit) on Linux 3.13 x86_64. Different algorithms are always tested on the same randomly generated scenario before comparing results. Runtime evaluations are based on Java's nanotime clock and performed for each algorithm individually excluding preliminary steps like graph population and building of the adjacency matrix. Computation of the latter takes around 250 milliseconds, for standard settings in the *Parking* and *Charging* scenarios, respectively. Note that all cost-optimal paths were computed using Dijkstra's algorithm. Choosing a different routing algorithm or employing a speed-up technique would yield the same benefit for all compared approaches. Modifying the path computation algorithm is an easy task, however, on a city scale (which the applications require) this would hardly yield any computational benefit. We present experiments for two realistic applications:

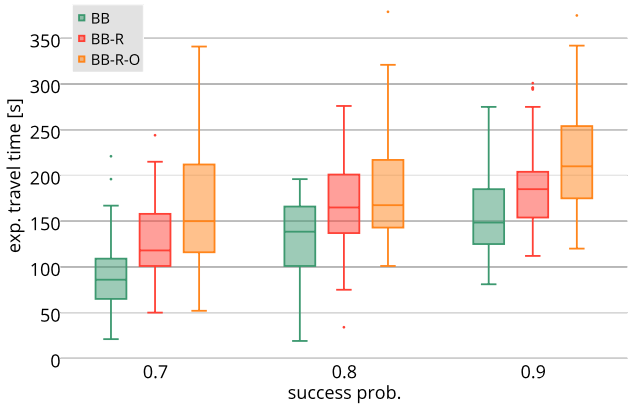
- *Parking* scenario (located in Bamberg, Germany): Given a probability threshold, we provide a route along parking spots which surpasses the threshold and minimizes the expected travel time. This scenario is based on ground truth extracted from OSM metadata.
- *Charging* scenario (located in Brussels, Belgium): Given a query position and a range limit (as used by electric vehicles), we provide a route along charging stations not exceeding the range limit and maximizing the success probability.

Note that these scenarios are complementary w.r.t. the criterion which is bounded and the criterion which is to be optimized, as explained in Section 4.1. Besides, while *Charging* relies on an hard numeric bound (distance), *Parking* relies on the more sophisticated expected value bound. Therefore we choose *Parking* as our main scenario. We will not present all charts for both scenarios, however noting that corresponding charts show the same behavior.

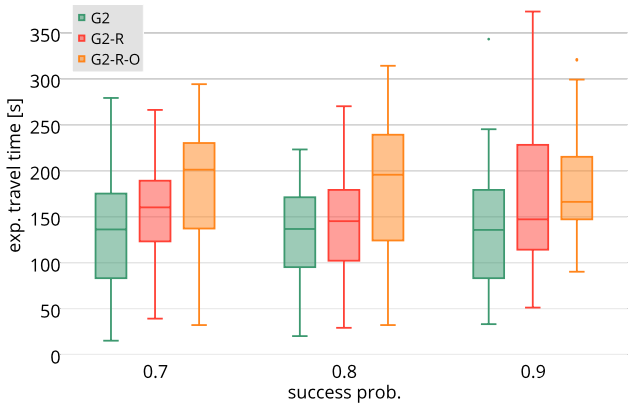
6.1 Parking Scenario

We generated the following test cases on the road network of the city of Bamberg, Germany, containing approximately 10.000 nodes and 20.000 edges as well as nearly exhaustive metadata regarding parking spots. For every test case, a target node is randomly drawn from all road network nodes of degree ≥ 1 within a three kilometer radius from the city center. Then, an isochrone of 800 meters walking distance is computed around the target. Let N be the number of resources (according to the ground truth) within the isochrone. In our experiments, resources are rather dense, i.e., $25 \leq N \leq 100$. Subsequently, the query node q is randomly drawn from all nodes within the isochrone. This corresponds to the use case where we expect the user to trigger the query when they are in the vicinity of their target. The average and maximal distance from q to a resource are by construction 800 and 1600 meters, respectively. Finally, $M \leq N$ observations of resource availability are randomly distributed among the resource locations, and the respective sojourn times in the states `available` and `consumed` are set. For reasons of clarity, in our experimental settings the sojourn times are set to the same configurations for all resources. We assume the expected time a spot stays vacant (`available`) to be 3 minutes and the expected time a spot stays occupied (`consumed`) to be 90 minutes. Note that the resources could easily be parametrized separately to model differently volatile resources. In this scenario, a probability threshold is given, and as a cost function we use travel time as formalized in Definition 3. The optimal resource route is the one with the least expected travel time among all resource routes with a success probability exceeding the threshold.

[1] http://www.openstreetmap.org/

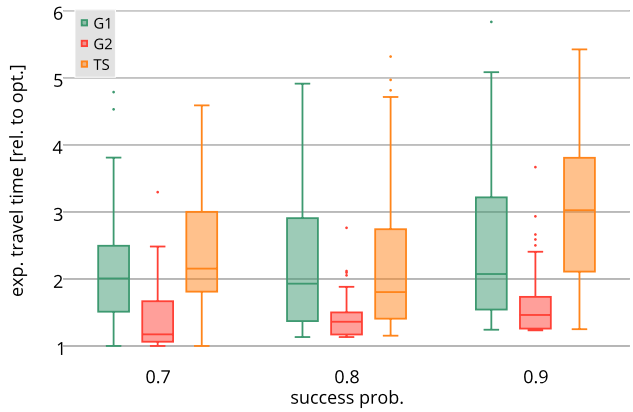


(a) **BB**-related algorithms

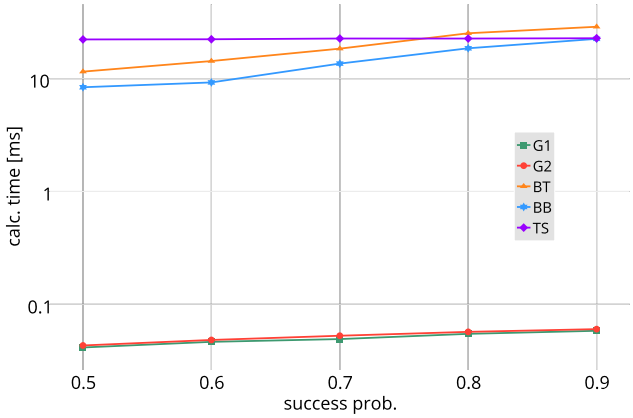


(b) **G2**-related algorithms

Figure 2: Illustration of the influence of model complexity on the quality of results (*Parking* scenario.



(a) Expected travel time relative to optimal solution



(b) Calculation time

Figure 3: Illustration of quality as well as efficiency of all algorithms in the *Parking* scenario.

First, we want to evaluate how much the additional information held by our probabilistic model improves result quality. Recall, that our model supports reappearance and incorporates short term observations, two properties that distinguish this work from others. In order to prove that the gain in result quality outweighs the gain in model complexity, we trim our algorithms **BB** (branch and bound) and **G2** (greedy approach with probability per cost heuristic) to partially ignore the information provided by the underlying model. In a first step, we disable the possibility of resource reappearance, we denote these approaches by **BB-R** and **G2-R**, respectively. This means, **BB-R** and **G2-R** proceed like their respective counterparts but do not revisit resources which have previously been observed as `consumed`. This corresponds to a simpler probabilistic model without the feature of resource reappearance. In a second step, we additionally disable short term observations. We denote these approaches by **BB-R-O** and **G2-R-O**. As before, they proceed like **BB-R** and **G2-R**, respectively, but additionally ignore any short term observations. Hence, the variations emulate an even simpler model which only allows static uncertainty, as used in [4], for example.

The results for the **BB**-related and the **G2**-related algorithms are shown in Figures 2(a) and 2(b), respectively. Both figures depict the same settings. It is obvious that requiring a greater probability threshold results in resource routes with longer expected travel time. Therefore, the overall increase in expected travel time is consequential. Both figures clearly show that the algorithms which rely on greater information, i.e., use a more complex model, yield better results. Figure 2(a) visualizes the results of the branch and bound approaches which are optimal w.r.t. to the information available. As claimed, **BB** on average outperforms **BB-R**, its counterpart which does not allow reappearance by at least 20 percent. **BB-R**, in turn, outperforms its counterpart which does not incorporate short term observations, **BB-R-O**. This supports the previously made claim that resource reappearance and short term observations do indeed improve the quality of results. From Figure 2(b) we observe, that simpler algorithms also benefit from the additional information contained in the model. Comparing the two figures, **BB**-algorithms of course yield better results than **G2**-algorithms and do so with significantly less variance than the greedy approaches. This is because the heuristics rely on chance in the form of beneficial problem settings in order to generate near-optimal results.

Next, let us investigate the performance of the algorithms presented. As mentioned before, there exists no work which is fully comparable. However, as the PRRQR is related to the TSP and clustering is commonly used to approximate the TSP (as in [3]), we use this concept to implement an approximative comparison partner denoted by **TS**. It is important to mention that **TS** does not support resource reappearance, because otherwise the heuristic would

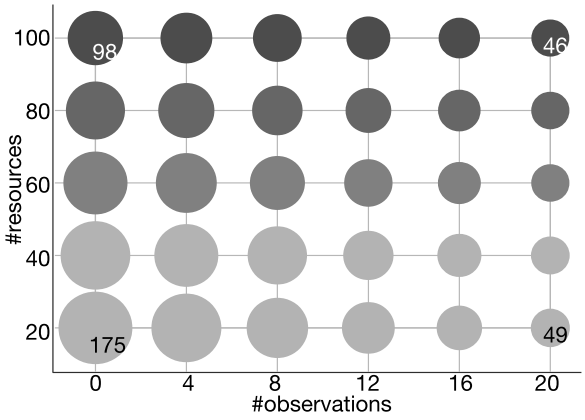
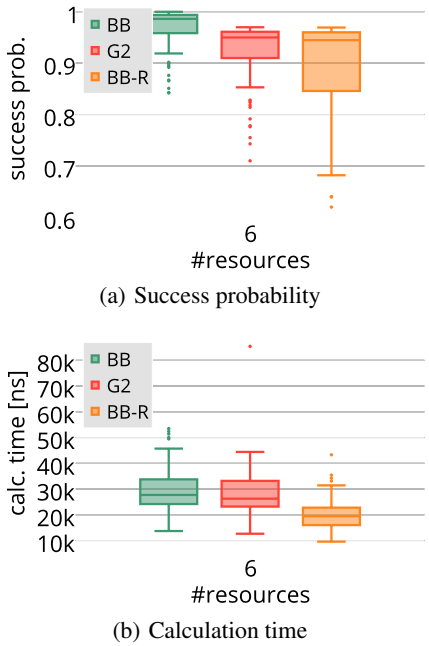


Figure 4: Influence of the number of resources and the number of observations on the expected travel time, i.e., result quality (for a probability threshold of 0.7)

not visit sufficiently many distinct resources to achieve a comparable success probability. Before we present the results, let us explain how **TS** proceeds. In a first step, **TS** conducts a k-medoid clustering on the set of all resources, where experimentally $k = 6$ has proven adequate. Subsequently, a TSP on the cluster medoids (starting at the query node) is solved. Then follows the actual resource route computation. It starts at the query node and computes the cost-optimal path to the first medoid. In the respective cluster, a greedy depth-first search (starting at the medoid) is conducted, returning an approximation of the cluster-internal cost-optimal path. From the last resource of the cluster we compute the cost-optimal path to the next medoid. This procedure is continued until the resource route exceeds the given probability threshold. **TS** serves as an algorithmic competitor based on a simpler probabilistic model but with a solid heuristic that has proven efficient when solving TSP-related problems. Note that the cost-optimal paths between all resources are precomputed in order to make the comparison to our algorithms – which use the precomputed adjacency matrix – fair.

We compare **TS** to all algorithms introduced in Section 5, i.e., the two greedy approaches **G1** and **G2** as well as the exact solutions **BT** and **BB**. Figure 3(a) shows the quality of the results produced by the approximative algorithms, i.e., **G1**, **G2**, and **TS**. Their respective expected travel times are given relative to the optimal results. The higher the probability threshold, i.e., the more complex the task, the greater the discrepancy between optimal results and approximation. Although **G2** relies on the rather simple probability-to-cost ratio heuristic, it significantly outperforms its comparison partners. While **G2** yields near-optimal results in the easier settings, the optimal solutions in the most elaborate scenario (probability threshold 0.9) undercut its expected travel times on average by about 30 percent. This gain in quality, however, comes at the price of calculation time, as depicted in Figure 3(b). This illustration shows the averaged runtimes of all algorithms when increasing the required probability threshold. The greedy approaches generate results in almost interactive time, while **BB**, **BT**, and **TS** are around two to three orders of magnitude slower. However, it is important to note that two orders of magnitude only correspond to around 100 ms of calculation time. Comparing the exact algorithms, we observe that **BB** outperforms **BT** which can be attributed to the forward estimation. The competitive approach **TS** performs in constant time of about 150 milliseconds (for the same number of resources), however generating the worst results.



(a) Success probability

(b) Calculation time

Figure 5: Illustration of quality as well as efficiency of selected algorithms in the *Charging* scenario.

Finally, we want to explore how volatile the results are w.r.t. the model parameters. We restrict ourselves to the optimal solution provided by **BB**, seeing as the quality ratio of optimal to approximative solutions has been explored above. Figure 4 depicts the influence of the number of parking spots relative to the number of short term observations of vacant parking spots. Thus, each circle in the plot corresponds to a pair of parameter values, and the diameter of each circle represents the average expected travel time of this scenario in seconds, as do the numbers in the corner circles. The result shows the expected behavior that with an increasing number of parking spots, expected travel time decreases. Furthermore, for any given scenario, it can be seen that the increased amount of short term observations also reduces the expected time until a vacant spot is found. Similarly expectable behavior is observed when varying the sojourn time parameters $1/\lambda$ and $1/\mu$, therefore further charts are omitted.

6.2 Charging Scenario

For *Charging* we generated test cases on the road network of the city of Brussels, Belgium, containing approximately 30.000 nodes and 67.000 edges. For every test case a query node is randomly drawn from all road network nodes of degree ≥ 1 within a 6 kilometer radius of the city center. Then, an isochrone of 6 kilometers is computed around the query node, wherein 6 resource locations are randomly drawn. We have evaluated other numbers of resources but the results do not reveal additional information and are therefore omitted here. Compared to *Parking*, where nearly every street holds at least one resource, this scenario models resource scarcity. Again, if N denotes the number of resources (6 in our experiments), then $M \leq N$ observations of resource availability are randomly distributed among these resource locations. The expected time a charging station remains vacant (`available`) is set to 30 min, and the expected time it remains occupied (`consumed`) is set to 50 minutes. As before, every charging station may be

parametrized individually, however we pass on it for reasons of clarity and lack of ground truth. In this scenario an absolute distance bound of 6 kilometers is giving, emulating the remaining range of an electric vehicle with low battery. Note that in contrast to *Parking*, this bound is strict and cannot be exceeded. Every algorithm computes a route with an absolute distance of 6 kilometers, the optimal resource route is the one with maximal success probability.

In a first setting, we compare the result quality of our exact algorithm **BB**, our greedy solution **G2** and **BB-R**, the branch and bound variation which does not incorporate resource reappearance. Additionally to the scarcity of resources, the remaining range (6 kilometers) is only double the average distance from query node to the next resource (3 kilometers, as resources are distributed uniformly within the isochrone). Due to these tightened constraints, superiority of the optimal results generated by **BB** becomes more apparent. In almost three out of four runs, **BB** yields a success probability of over 95 percent, outperforming **G2** significantly. While the greedy heuristic worked well before, it is now easily lead down a considerably less beneficial branch of the search tree. Nonetheless, **G2** still produces slightly better results than **BB-R**. Again, this advocates our model which supports resource reappearance. Even an approximative approach on our model yields better results than an exact algorithms on a less sophisticated model due to lack of information. Of course, a simpler model needs less intricate function evaluations. In our case, however, the difference is merely a matter of microseconds, as depicted in Figure 5(b).

Concludingly, we have empiricially proven the benefit of our probabilistic model. It improves the quality of results by incorporating richer information, especially for complex but also for simpler tasks while not causing any significant computational overhead. On the contrary, our greedy approaches deliver competitive results in near-interactive time while our branch and bound approach yields optimal solutions in efficient time.

7. SUMMARY AND OUTLOOK

In this paper, we investigate probabilistic route queries in road networks where the user is guided along a set of resources in order to maximize the probability of encountering an available resource. We aim to find a route with minimal expected cost among all routes exceeding a given probability threshold. We propose a novel framework in which resources are modeled as continuous-time Markov chains with two states, `available` and `consumed`. In contrast to similar problems, our framework allows for consumed resources to reappear and takes short term as well as long term observations into account. The introduced query, referred to as PRRQR, is theoretically NP-complete and has an unlimited search space.

To solve this problem, we propose approximative as well as optimal solutions. We employ two different search heuristics in a greedy algorithm to achieve a trade-off between accuracy and calculation time. Furthermore, solutions using backtracking and a branch and bound approach provide optimal solutions in competitive time. We demonstrate the superiority of our model as well as the efficiency and effectiveness of our algorithms on two realistic applications. The first is the search of a vacant parking spot, and the second is the search for a vacant charging station for electric vehicles.

For future work, we want to turn to settings considering other types of observations like competing drivers looking for the same type of resource. Furthermore, we want to investigate the influence of edge costs which might change during the search.

8. REFERENCES

[1] H. Chen, W.-S. Ku, M.-T. Sun, and R. Zimmermann. The multi-rule partial sequenced route query. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACMGIS'08)*, pages 10:1–10:10, 2008.

[2] D. Cheng, M. Hadjieleftheriou, G. Kollios, F. Li, and S.-H. Teng. On trip planning queries in spatial databases. In *Proceedings of the 9th International Conference on Advances in Spatial and Temporal Databases (SSTD'05)*, pages 273–290, 2005.

[3] A. Delis, V. Efstathiou, and V. Verroios. Reaching available public parking spaces in urban environments using ad hoc networking. In *Proceedings of the 12th International Conference on Mobile Data Management (MDM'11)*, pages 141–151, 2011.

[4] N. Dolev, Y. Doytsher, Y. Kanza, E. Safra, and Y. Sagiv. Computing a k-route over uncertain geographical data. In *Proceedings of the 10th International Conference on Advances in Spatial and Temporal Databases (SSTD'07)*, pages 276–293, 2007.

[5] G. Gidofalvi, T. B. Pedersen, T. Risch, and E. Zeitler. Highly scalable trip grouping for large-scale collective transportation systems. In *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '08, pages 678–689, New York, NY, USA, 2008. ACM.

[6] F. Graf, H.-P. Kriegel, M. Renz, and M. Schubert. MARiO: Multi attribute routing in open street map. In *Proceedings of the 12th International Conference on Advances in Spatial and Temporal Databases (SSTD'11)*, 2011.

[7] Y. Kanza, R. Levin, E. Safra, and Y. Sagiv. Interactive route search in the presence of order constraints. *The International Journal on Very Large Data Bases (VLDB'10)*, pages 117–128, 2010.

[8] 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 (SSTD'09)*, pages 153–170, 2009.

[9] M. Kolahdouzan, C. Shahabi, and M. Sharifzadeh. The optimal sequenced route query. *The International Journal on Very Large Data Bases (VLDB'08)*, 17(4):765–787, 2008.

[10] R. Levin, Y. Kanza, E. Safra, and Y. Sagiv. An interactive approach to route search. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACMGIS'09)*, pages 408–411, 2009.

[11] S. Ma, O. Wolfson, and Y. Zheng. T-share: A large-scale dynamic taxi ridesharing service. In *Proceeding of International Conference on Data Engineering (ICDE'13)*, pages 410 – 421, 2013.

[12] L. Matos, J. Nune, and A. Trigo. Taxi pick-ups route optimization using genetic algorithms. In *Proceedings of the 10th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA'11)*, pages 410–419, 2011.

[13] J. Norris. *Markov Chains*. Cambridge Univ. Press, Cambridge, 1998.

[14] D. O. Santos and E. C. Xavier. Dynamic taxi and ridesharing: A framework and heuristics for the optimization problem. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*, pages 2885–2891, 2013.