Short Paper

Dynamic Resource Routing using Real-Time Information

Sebastian Schmoll
Ludwig-Maximilians-Universität
Munich, Germany
schmoll@dbs.ifi.lmu.de

Matthias Schubert
Ludwig-Maximilians-Universität
Munich, Germany
schubert@dbs.ifi.lmu.de

ABSTRACT

Searching the nearest available resource is an important query with many applications in spatial database systems. Examples are searching free parking spots or charging stations in a road network. Information about the availability is usually approximated as long-term statistics. Recently, online systems and sensor networks become more and more common providing access to real-time information about resource availability. In this paper, we consider searching the next available resource in a road network considering real-time information about all resources in a target area. To make the best use of this information, it is not enough to predict a static result route. Instead we propose to model the problem as a Markov decision process (MDP) and compute a routing policy which allows flexible reaction to newly observed developments. In our experiments, we demonstrate that our new approach is capable to exploit the additional information and outperforms previous approaches built on long-term distributions on a real-world parking data set.

1 INTRODUCTION

In modern location based services, queries often aim at finding a spatial resource of a specific type. In contrast to a point of interest (POI), a spatial resource is not generally available but might be occupied at a certain point in time. Correspondingly, an occupied resource might become available during search time. Examples for spatial resources are parking spots, charging stations, rental vehicles or drop-off locations for rental vehicles. To handle the availability of spatial resources, online systems can provide real-time information about currently available resources. However, there is no guarantee that a currently available resource will be available at the arrival time of our user. One might argue that a reservation service might solve any such problem, but reservation services often add additional complexity which often makes them impractical. Reserved resources might be used without authorization and generally available resources might remain unused due to just-in-case reservations. For example, a parking spot might be occupied even if it was reserved before because the previously parked vehicle was not removed in time. Correspondingly, inner-city parking spots might remain unused due to rich people having a permanent reservation just in case they want to go into town. To conclude, for many applications reservation systems might be unreliable and expensive to enforce. Thus, we focus on the case that a resource is claimed by the first user to arrive.

We investigate the task of finding an available resource spending minimal travel time or any other type of cost. Furthermore, we assume real-time information on all available and occupied resrouces in a specified target area. To properly exploit the available real-time information, a simple route is not enough. Since the availability of resources might be constantly changing, a fixed route cannot include reactions to developments during the search. Thus, we need to compute a policy which indicates the most promising action for each situation during the search. For example, when looking for a parking spot a natural behavior is to take any parking spot getting available along the way. A resource route as proposed in [5] would follow its fixed search order which is based on the information provided at search time. Thus, a resource route would only consider the vacant spot if it is scheduled in the route. In contrast, a policy driven system would recognize the improved situation and propose to take the available parking spot if it is a suitable location.

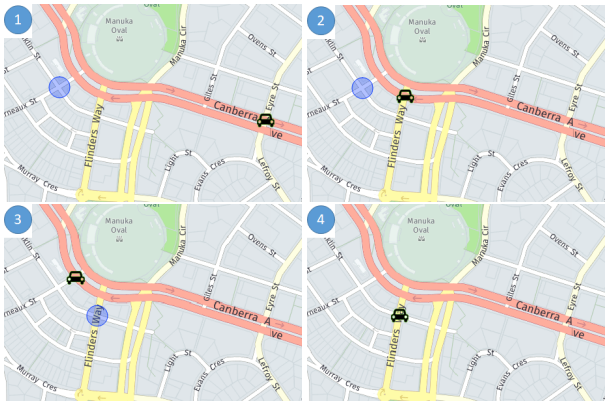

Figure 1: Example for a parking search using our new algorithm. The car symbol denotes the car whose driver is looking for a parking spot. The blue circles denote at least one free available parking slot at the corresponding location.

In this paper, we propose to employ the statistical model of Markov decision processes (MDP) to learn such a policy. A policy is a mapping of any possible situation to the most promising action which an agent should perform to maximize the probability of success. Thus, our proposed solution allows that resources might get occupied or available during the search. The availability of each resource is modeled as a time-continuous Markov chain. Based on these probabilities, we define an MDP to model resource search with real-time information. A drawback of this model is that the state space grows exponentially with the number of acceptable resources. Since in many cases resources are stacked at particular areas, we propose resource aggregation to limit the number of considered states. Additionally, we propose to precompute policies for relevant target areas and store them in a data structure for efficient retrieval of the proposed action. To conclude, the contributions of this paper are as follows:

- A model to compute a search policy for resource search considering real-time information.
- A method for precomptung policies and apply the learned policy for guiding a user.

- Experiments on real-world parking data demonstrating the advantage of using real-time information.

The rest of the paper is organized as follows: Section 2 surveys previous works on resource search and MDPs. In Section 3, we give a brief introduction to MDPs and introduce our model for resource search. Section 4 outlines the use of policies for routing services. In our experimental evaluation in Section 5, we demonstrate the advantages of our new approach. Section 6 concludes the paper with a short summary and directions for future work.

2 RELATED WORK

The methods being closest to our approach are [5] and [3]. [5] describes a method for computing a route minimizing the expected search time in a setting where resources can get occupied and available during the search. However, real-time information is only considered to be available at query time. Thus, the resulting resource route cannot react to updates during the search. In [3], the authors compute a policy for resource search. The solution is basically an MDP and the proposed method resembles the basic value iteration algorithm for computation. The major difference to our work is that the method does not consider the availability of real-time information during the search. However, there are already several prototypes and testbeds providing this type of information for urban areas such as the city of Melbourne in our evaluation. Our method employs Markov decision processes (MDP) as described in [7]. Though there is a plethora of more sophisticated solution methods for computing MPDs like [1, 2, 4, 6, 8, 9], we employ the basic value iteration algorithm since the approach in this paper precomputes policies and afterwards queries these policies to guide a user.

3 PROBLEM SETTING

3.1 Markov Decision Processes

A Markov Decision Process consists of a set of states S, sets of actions $\forall s \in S : A(s)$, the probability distribution for each state action pair $P(s'|s,a)$ and a cost (or reward) function $c(s,a)$. It is important to note that $P(s'|s,a)$ indicates that the result of the action (s,a) is uncertain and that any state transition $s \rightarrow s'$ with $P(s'|s,a) > 0$ for any $a \in A(s)$ is potentially possible.

A policy π is a set of state action pairs (s,a) that determines for any state $s \in S$ exactly one action $a \in A(s)$. In our usage of MDPs, we assume terminal states S^* which indicate the targets of the search. Thus, the goal of a policy π is to reach any terminal state $s^* \in S^*$ and a proper policy guarantees that we will reach a terminal state s^* at some point in time. To compare policies, we now define the expected cost $U(s)$ which is also referred to as utility or value function in literature.

In general, $U^\pi(s)$ of policy π represents the expected cost which is aggregated over all possible state sequences starting with state s. We define the optimal policy π^* for any state s as:

$$\pi^*(s) = \operatorname*{argmin}_{a \in A(s)} \sum_{s'} P(s'|s,a) \cdot U^*(s')$$

Whereas $U(s)$ can be computed by the following equation also known as Bellman equation:

$$U(s) = \min_{a \in A(s)} c(s,a) + \gamma * \sum_{s'} P(s'|s,a) \cdot U(s')$$

In other words, the minimal expected cost for state s are achieved by taking the action a where the sum of direct costs $c(s,a)$ and the expected future cost are minimal. The parameter γ is used to weight direct costs against future costs which is often beneficial for the convergence of computation.

3.2 Modelling Resource Search

After giving a brief general introduction to MDPs, we will now turn to modelling resource queries as an MDP problem. In our setting an agent, e.g., a driver or a person moves in a road network. A road network $G = (N, E)$ consists of a set of nodes N and a set of edges $E \subseteq N \times N$. Each edge $e \in E$ yields traversing costs c_e, e.g., travel time or distance. Additionally, we model resource locations as nodes $P \in N$. For each resource $p \in P$, the boolean function $a(p)$ denotes 0 if p is occupied and 1 if p is available.

The state space of our MDP is a concatenation of the agent position $l \in N$ and the value of $a(p)$ for each $p \in P$. Thus, the number of all possible states is $|N| \cdot 2^{|P|}$. All states where $l \in P$ and $a(l) = 1$ are considered as terminal states because the agent arrives at the location of a currently available resource. The actions in our model consists of the possible routing directions at each crossing, e.g., turn left, turn right, go straight or turn.

After defining states and actions, we now have to provide the likelihoods $P(s'|s,a)$. In our setting, the next location of the agent l' is determined by the selected action a. However, the change of the available resources depends on the status changes of the resources P. Thus, we need to compute the likelihood that a resource p changes its current status from occupied to available or vice versa. Given that we assume exact information at the current point of time, we only need to estimate the distribution of a state change for the required time to travel from l to l'. To do so, we rely on the continous time Markov model being proposed in [5]. The model predicts the likelihood $P(p(t) = 0)$ that resource p is available at time t based on two parameters, the average occupation time and the average vacancy time. Both parameters can be easily derived from empirical data.

Finally, the cost of each action (s,a) is considered as the travel cost $c_{(l,l')}$ for the edge (l,l') where l is the agent location in state s and l' is the node after taking action a. Obviously, the given model allows to compute proper policies as long as any resource does not converge to a state of being constantly occupied. In this case, returning infinitely often to the resource will result in finding the resource available at some point in time.

After defining all components to model resource search as an MDP, we now want to briefly describe the employed value iteration algorithm for computing the minimal expected cost $U^*(s)$ for each state $s \in S$. The idea of value iteration is to employ the Bellman equation to each state in each iteration until the utility for any state does not improve significantly anymore. The pseudcode of value iteration is presented in algorithm 1. After computing $U^*(s)$ processing the optimal policy is straight forward by applying the *argmin* function instead of the *min* function in the Bellman equation. Note that we employed value iteration due to its simplicity and for comparability with previous approaches. For a more detailed introduction to value iteration please refer to [7].

4 USING MDPS FOR RESOURCE SEARCH

A general problem of the method proposed above is the exponential increase of the state space with the considered resources P. Obviously, computing the basic value iteration method for large numbers on resources in not feasible. Thus, in order to make the approach applicable for real systems, we need to make sure that

Algorithm 1 Value iteration algorithm for computing the expected cost U of an optimal policy.

```
1:  procedure VALUE ITERATION(mdp, ϵ)
2:      init U′
3:      repeat
4:          U ← U′
5:          δ ← 0
6:          for all s ∈ S do
7:              U′[s] = min_{a∈A(s)} c(s, a) + γ ∗ Σ_{s′} P(s′|s, a) · U[s′]
8:              if |U′[s] − U[s]| > δ then
9:                  δ ← |U′[s] − U[s]|
10:     until δ < ϵ(1 − γ)/γ
11: return U
```

the number of potential resources is as small as possible. Fortunately, in many cases the region where resources are acceptable is usually limited to a certain spatial region being specified by a user. For example, parking spots should be in walking distance to the actual target of the user. Similarly, charging stations should be close to planned route and in the region where the battery is already empty to make charging feasible. Similar statements hold for rental vehicles such as bikes and drop off stations. Thus, a first approach to limit the considered resources would be to compute isochrones around the actual target of the user and only consider resources within the tolerable walking distance. However, this approach would not work if the user would not specify an exact target location but only a rough area where he needs the resource. Additionally, in many cases the amount of acceptable resources would still push the computational effort beyond interactive query times. Thus, we propose another approach where MDPs are precomputed for typical target regions. For example, parking MDPs might be precomputed for typical shopping areas and MDPs for bike sharing make sense around universities. Let us note that precomputation can be done even for overlapping areas. Finally, the selection of the most suitable area can be easily performed by the user based on map interface displaying the available areas.

However, limiting the spatial area might not yield a successful reduction of available resources because many types of resources are spatially stacked on close-by locations. Examples are parking spots and rental bikes. For these type of resources, we propose to aggregate close-by resources and model them as a single aggregate resource in the MDP. For example, instead of modeling a road segment with ten parking spots as ten nodes having one resource, we aggregate all spots on the segment into a single aggregate resource. Of course, we have to adapt the probabilities for computing the likelihood $P(p(t) = 0)$ to mirror the fact that it is enough that any of the underlying resources is available. Thus, the probability $P(\hat{p}(t) = 0)$ of the aggregate resource $\hat{p}$ for being available corresponds to the likelihood that not all of the underlying resources are occupied at time t. Formally, given a set of close-by resources $\hat{P} = \{p_1, \ldots, p_k\}$ which are aggregated to the aggregate resource $\hat{p}$ then $P(\hat{p}(t) = 0) = 1 - \prod_{p_i \in \hat{P}} P(p_i(t) = 1)$. However, $P(\hat{p}(t) = 1) = \prod_{p_i \in \hat{P}} P(p_i(t) = 1)$ because $\hat{p}$ can only be considered as occupied if none of the underlying resources is available. A drawback of resource aggregation is that it removes the time for traveling between the aggregated resources. Thus, the learned policy might not be able to handle searches within the combined resources. However, we argue that with a proper selection of resources, the user should be able to solve the problem by himself, e.g. finding a free resource among the combined resources. A heuristic, we employed to find sets of resources that can be aggregated is to make sure that all heuristic can be reached by making the same routing decision. In other words, all resources are placed on the same edge of the underlying road network.

A final aspect of our proposed method is the use of the learned policy in systems for guiding users to an available resource. As mentioned before, we precompute policies and let the user decide on the best suitable precomputed query region in order to find a suitable policy. Afterwards the system can combine the current user location and the online information about available resources to determine the current state and propose the direction aka action being stored in the policy. If the user wants to have an outlook on the following actions, the system could compute the most likely states for the current situation. However, in a volatile environment it is likely that these directions will change during travel to the next waypoint. A remaining task in this solution is to efficiently retrieve the optimal action for the current state for a given policy. Though we can identify the current state by combing location and resource information, we still have to find the state in the precomputed policy. Given the exponential size of the state space finding the entry for a particular state has to be done in an efficient way. Thus, we propose to store policies in a two dimensional array. We index the set of locations N by enumeration which indicates the first dimension. For the index of the second dimension, we encode the status information of the considered resources. This is done by mapping each resource to a particular bit in a bit vector of length $|P|$ and set the bit according to the availability of the resource. The resulting bit vector is then interpreted as an integer which indicates the second index in the array. Thus, the proposed action is retrieved in constant time.

5 EVALUATION

In order to evaluate our approach on a realistic scenario, we simulate the states of the parking spots for the city of Melbourne. The simulation samples for a given time-stamp and time interval the next states of any given parking spot. We have developed two different simulation models. The first model is based on the continuous-time Markov chain (similar to our MDP model) which enables us to do experiments based on the same conditions that our MDP assumes. However, continuous-time Markov chains might not provide a suitable model for real-world data. Thus, we implemented a simulator that is based on a real-world parking dataset, namely the freely available Melbourne "Parking bay arrivals and departures 2014" dataset[1]. The dataset that contains arrival and departure times of most central business district parking bays. When we take the arrival and departure time into consideration, we can decide for every contained time-stamp, whether the parking bay is available or occupied. This simulation can be equated to a realistic real-time information system because it retrieves events that actually happened at the given time-stamp.

We compare our new approach denoted as D3RI to the UGCM algorithm that was proposed in [3] which also computes a policy but does not consider real-time information.

In figure 2, we can observe that the average time of finding a parking spot in the simulator is two to four times shorter

[1] https://data.melbourne.vic.gov.au/Transport-Movement/Parking-bay-arrivals-and-departures-2014/mq3i-cbxd

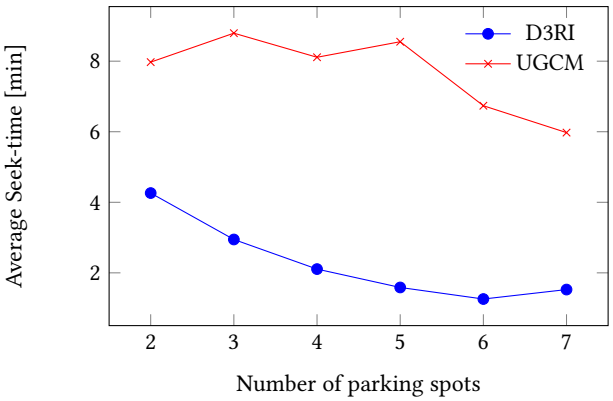

Figure 2: This figure shows the average search time of the agent in the time-continuous Markov chain simulator. Therefore we ran 1000 simulations and computed the average parking search time.

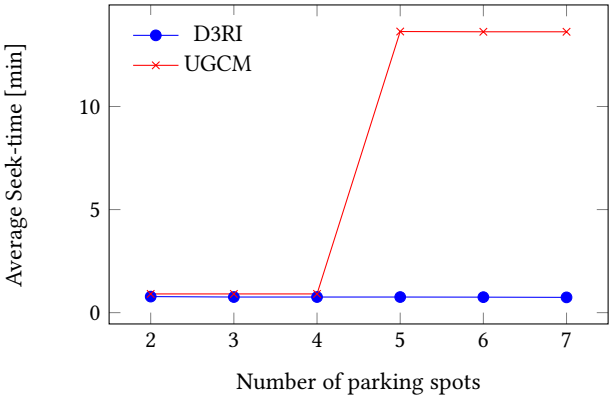

Figure 3: This figure shows the average search time of the agent in the Melbourne simulator. Therefore we ran 10 simulations on different days and computed the average parking search time.

with the approach proposed in this paper D3RI than the UGCM approach[3].

This is mainly because D3RI considers the real-time information of the parking spots while UGCM does not know if the parking spot, the agent is currently heading to, is available. This property can be observed in the real-world (Melbourne parking bays) simulation as well (c.f. figure 3). By knowing the exact states of the parking spots, D3RI can guide the driver directly to a freely available parking bay. However, if the information is not available, the driver may head to a nearby currently occupied parking spot. Because there are more possibilities for occupied parking spots, the approach of [3] becomes even worse when there are more spots available, since UGCM does not know which of the parking slots are free right now and usually with increasing amount of parking spots the amount of occupied spots increases as well. As expected, the use of real-time information yields a strong advantage which can be exploited by the D3RI model presented in this paper.

Figure 4 illustrates the runtime of our approach regarding the number of parking spots. In this experiment, we have used an error acceptance rate ϵ of 0.001 and a γ-value of 0.99. Obviously, the runtime increases exponentially with the amount of

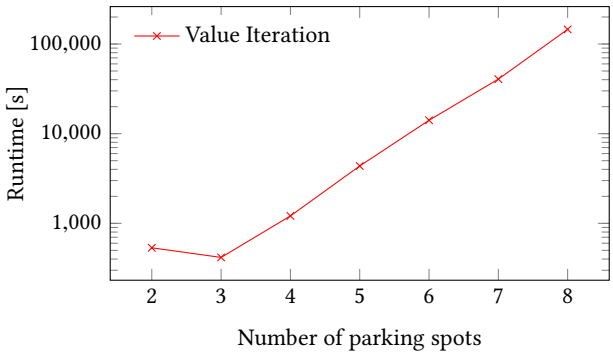

Figure 4: This figure shows the runtime of our approach depending on the number of spots. One may pay attention to the logarithmic y-axis.

considered parking spots. This suggests how essential resource aggregation is to reduce the computational overhead and the size of the policy. To conclude, without resource aggregation modeling realistic scenarios quickly get infeasible.

6 CONCLUSION

In this paper, we propose a novel approach for dynamic resource search in spatial networks considering real-time information on resource availability. To exploit valuable real-time information, solutions must react to recent developments in a flexible way. Therefore, we propose to learn a routing policy which provides the most promising action for any potentially occuring situation. To compute such a policy, we model resource search using real-time information as a Markov Decision Process (MDP). Furthermore, we discuss the design of a query system using precomputed policies for guiding a user to an available resource. In our experiments, we compare our new approach using real-time information to a previous approach which computes policies based on historical data only. The results indicate that our new approach can exploit the real-time information to clearly outperform the comparison partner. For future work, we plan to examine techniques to mine areas for which building an MDP is most valuable. Furthermore, we will examine techniques to improve resource aggregation in order to compute MDPs for larger areas. Finally, we plan to examine database technology for storing and querying policies with very large state spaces.

REFERENCES

[1] Andrew G Barto, Steven J Bradtke, and Satinder P Singh. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72, 1 (1995), 81–138.
[2] Blai Bonet and Hector Geffner. 2003. Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming.. In *ICAPS*, Vol. 3. 12–21.
[3] Qing Guo and Ouri Wolfson. 2016. Probabilistic spatio-temporal resource search. *GeoInformatica* (2016), 1–29.
[4] Eric A Hansen and Shlomo Zilberstein. 2001. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence* 129, 1-2 (2001), 35–62.
[5] G. Jossé, K. A. Schmid, and M. Schubert. 2015. Probabilistic Resource Route Queries with Reappearance. In *Proceedings of the 18th International Conference on Extending Database Technology (EDBT15)*.
[6] H Brendan McMahan, Maxim Likhachev, and Geoffrey J Gordon. 2005. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *Proceedings of the 22nd international conference on Machine learning*. ACM, 569–576.
[7] Stuart J. Russell and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach* (2 ed.). Pearson Education.
[8] Scott Sanner, Robby Goetschalckx, Kurt Driessens, Guy Shani, et al. 2009. Bayesian Real-Time Dynamic Programming.. In *IJCAI*. Citeseer, 1784–1789.
[9] Trey Smith and Reid Simmons. 2006. Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic. In *AAAI*. 1227–1232.