A Framework on Spatio-Temporal Resource Search

Qing Guo, Ouri Wolfson, and Daniel Ayala
Department of Computer Science
University of Illinois at Chicago
qguo@cs.uic.edu, wolfson@cs.uic.edu, dayala@uic.edu

***Abstract*—In this paper, we establish a framework for the spatio-temporal resource search problem. In a spatio-temporal resource search problem, a mobile agent tries to find a desired static resource in a network. The resources are statically located in the network. The task is to guide the agent through the network to find a resource with lowest possible cost. There are lots of applications in urban transportation systems that fit in this kind of setup. One example is a vehicle that searches for street parking in a neighborhood of a city. Another example is a taxicab that cruises in an area to find customers. In these examples, street parking spaces and taxicab customers are the static resources being searched. In this work, we develop two algorithms, namely the Random Walk Algorithm and the Prophet Algorithm, that serve as the lower and upper bounds on the performance of any reasonable algorithm that assumes some kind of information about the availability of the resources. Furthermore, when assuming that the resource-availability probabilities for each edge in the network are known to the agent, we develop the Probabilistic Algorithm using these probabilities. We conducted experiments for the three algorithms under different conditions, and the results show that using the Probabilistic Algorithm, the agent can perform much better than having no information at all, and surprisingly close to the performance upper bound.**

***Keywords*—spatial databases, probability.**

I. Introduction

In this paper, we present a framework that deals with mobile agents and statically located resources. Each resource can be used by a single agent at a time. The agent searches for one of the resources that are located in a network. During a search, an available resource might become unavailable because it is obtained by some other agent, and vice versa.

This framework is an abstraction of many real-world problems. For example, in the situation where a vehicle driver is looking for a street parking space, the vehicle is the mobile agent, and the parking spaces are the resources. Another example is the situation in which a taxicab driver is cruising to pick up potential passengers. In this example, the taxicab is the mobile agent, and the passengers are the resources.

This framework has both spatial and temporal features, because the resources and the agent are restricted to a geographic road network, and the availability of the resources change over time. In fact, if the resource availability does not change over time, and assuming that the agent has full access to the availability information, then it becomes a trivial problem because the agent can simply go to a resource with the lowest cost and get it.

Our work is motivated by the increasing popularity of mobile devices and increasing accessibility to location-based services as a result. These developments make it possible to obtain the availability information about the resources in a spatio-temporal resource search. For example, in Ma et al. [1], end-user smartphones are used to detect potential customers for taxicab drivers; in a project called SFpark [2], wireless embedded sensors are used to detect available street parking spaces.

With the resource availability information, we are interested in developing methods that minimize the cost for the agent to find a resource. In the present work, the cost is merely measured by travel time of the agent for convenience. It is trivial to add other types of cost to the cost function, such as the walking distance from the street parking block to the final destination in the parking application.

In this work, we consider three levels of information about the resource availability that the agent has access to: no information at all, full current and future information, and partial information in a probabilistic form. These correspond to three algorithms we present in this paper. The *Random Walk Algorithm* assumes no information about the availability of the resources at all, and the agent can only depend on luck and performs a random walk to search for a resource. This corresponds to the way a search for a street parking space is currently conducted by an uninformed driver. The *Prophet Algorithm*, on the other hand, assumes that the agent knows exactly how many available resources there are at each location and at each point in time for now and in the future. This is an unrealistic algorithm that knows the future, and serves as a benchmark for the upper bound of performance.[1] The *Probabilistic Algorithm*, uses the probability that a location has at least one available resource.

The probabilistic data, as a form of uncertain data, is common in practice, and can be obtained from past data applying resource detection techniques that use crowdsourcing. An example is Xu et al. [3] where a crowdsourcing method is used to detect parking availability. Another example is in Mathur et al. [4], where the authors used existing municipal vehicles equipped with GPS receivers and ultrasonic rangefinders to detect street parking availability while driving by curbside parking spaces.

In our work, we assume that the static resources are located on the edges of a network, and the agent is able to travel through the network in search of a resource. Each edge, as a resource location, is associated with a probability of having at

[1]Performance is measured by the search-time until a resource is found.

978-1-4799-5344-8/15/$31.00 ©2015 IEEE

least one available resource. We further assume that the agent only makes decisions at the nodes. Also, by saying that an edge is *available*, we mean that the edge has at least one available resource. We use these two expressions interchangeably. The cost for an edge d with respect to the current location s of an agent, is defined as the shortest travel time from s to d.

Now we make an important comment concerning the Probabilistic Algorithm. We noticed that, it is possible to simply make the agent travel to an edge that is optimal in terms of probability and cost, and make her wait on that edge if it is unavailable when she arrives. This is the optimal strategy if the resources on that edge are uniformly distributed over time, e.g., 0.5 probability means that at each point in time, the chance of encountering an available resource is one half. However, in reality, even if an edge has a high probability, as soon as the agent traverses it and finds out that it is unavailable, it is very unlikely that the edge will become available immediately. For example, in the parking search application, vehicles tend to park for a certain period of time. To utilize this observation, we introduce the concept of *recovery function* used by the Probabilistic Algorithm. This means that when the agent discovers that an edge is unavailable, the probability of that edge drops to zero for a "recovery" period of time.

To test the resource search algorithms, we conduct experiments using real-world data from SFpark [2]. The results, as expected, show that the more information the agent has, the faster the agent can find a resource. Moreover, the introduction of recovery function is very useful for the Probabilistic Algorithm, making its performance improve dramatically. And surprisingly, the performance of the Probabilistic Algorithm is very close to the Prophet Algorithm. This means that the probabilities, as a type of partial information, are highly informative for the resource search problem, in the sense that their performance is close to the performance upper bound.

Here is a summary of major contributions of this paper:

- We establish the upper bound (obtained by the Prophet Algorithm) on the performance of any spatio-temporal resource search algorithm that uses only past and current resource-availability information (certain or uncertain).
- We compare different levels of information that an agent may use for the spatio-temporal search problem, and find that the probabilistic information is highly valuable in the sense that it comes close to the performance upper bound.
- We introduce the probability recovery function, which takes advantage of observations of the agent during a search to adjust probabilities dynamically.

Note that this paper has significant cost implications. Our results indicate that in terms of parking search effectiveness, the fixed (in-pavement) solution that provides deterministic information and the mobile sensor (ultrasonic on vehicles) solution that provides probabilistic information, are comparable.

Now consider the costs of the two solutions in the Fisherman's Wharf area. SFpark has 777 fixed sensors in the area, each costing $\$300$ to install and $\$170$ annually to maintain [2]. So just the maintenance cost is $\$132\,000$ annually. In contrast, by installing ultrasonic mobile sensors on existing municipal and parking-enforcement vehicles, a probabilistic database of parking availability can be obtained at a fraction of this cost. Specifically, assume the sensor-cost of $\$400$ per vehicle [4], and assume that five vehicles are sufficient to provide hourly data for each Fisherman's Wharf block. Then the mobile solution can be provided for $\$2\,000$.

The rest of the paper is organized as follows. In Section II, we review related work. In Section III, the assumptions and notation of the problem used throughout the paper are presented. In Section IV, we present three algorithms for the spatio-temporal resource search problem; each algorithm uses a different level of knowledge of resource-availability information. The experiments we conducted are presented in Section V, and the conclusions are discussed in Section VI.

II. Related Work

As location-based services become increasingly accessible, it is possible to develop new approaches for sensing and monitoring available spatial objects. For example, In Mathur et al. [4], the authors equipped special vehicles with GPS receivers and ultrasonic range-finders, and made them drive through a road network. When driving by curbside parking spaces, the sensors detect whether they are available or occupied. At the end, a map of parking availability can be generated using the data collected during the process.

Besides installing sensors, crowdsourcing methods using end-user smartphones have appeared for the problem of detecting the availability of parking spaces, because it is easy to implement, and inexpensive for larger scale (such as citywide) services. For example, by classifying mobility patterns of smartphone users [5] or using Wi-Fi signature matching [6], parking and unparking activities for individual smartphone users and hence vehicles can be detected. Using the information of users' parking and unparking activities, it is possible to estimate parking availability at block levels [3]. A challenge that crowdsourcing methods face is that the data they can get is usually uncertain. This is because not all vehicles that are involved in parking activities subscribe to the service that carries the crowdsourcing.

Because the parking problem is important and representative, there has been a line of research addressing it. We categorize these efforts into two kinds. One is the parking *assignment* algorithms. In this category, efforts are put on assigning vehicles to available parking spaces, and the objective is to minimize the overall cost to society. Parking space reservation systems fall into this category. Several such systems have been developed [7], [8], [9]. These parking assignment systems assume that users are socially responsible and willing to sacrifice their own interests in order to achieve common good. This is because in an optimal solution that minimizes the overall cost, some users may be able to deviate from that solution to achieve lower costs for themselves [10].

Therefore, the other method of addressing the parking problem, the parking *search* algorithms were developed. These systems assume that users are selfish and they compete for parking spaces. This approach is more realistic because

current parking systems are mostly competitive rather than reservation-based. Ayala et al. [11], [12], [13] developed an approach that copes with the parking problem in a competitive setting. Assuming that the agents are selfish and only care about their own individual interests, the parking problem was explored in a centralized modal and a distributed model. The above work assumes that the agent is informed in real time of the availability of resources at each location. However, as discussed earlier in this section, the more powerful crowdsourcing methods will most likely generate uncertain data.

There has been some research that uses uncertain data to conduct resource searches [14], [15], [16], [17]. For example, Safra et al. [15] considered the probability of a spatial object as the confidence value of some query, e.g., whether a restaurant is good according to the user's preference. Therefore, for a specific user, the truth value of each spatial object does not change over time.

Verroios et al. [16] and Jossé et al. [17] assumed that the agent can get the accurate and deterministic availability information of the resources in the beginning of a search (our Probabilistic Algorithm does not make this assumption). Then they assumed some probability decaying functions modeling the probability that a known available resource stays available after some time. Note that the notion of probability decay over time in these papers is different than the notion of probability recovery function in our work. In our work, we do not assume that the agent knows the exact availability of the resources a priori. The probability of a resource only changes when the agent visits it and finds that it is unavailable.

Safra et al. [15], Verroios et al. [16], and Jossé et al. [17] all used extensions of the traveling salesman problem as the solution concept, which is known to be NP-hard. Therefore, they proposed different techniques to produce approximate solutions. One disadvantage of modeling the resource search problem using the traveling salesman problem is that a Hamiltonian path does not revisit the same resource, although it may become available before the agent finds a resource elsewhere.

Yuan et al. [14] considered the accumulated probability of finding a resource for any given route from a starting location to a specific destination. Their algorithm can be used in our model, but it would require adaptation, resulting in a more complicated algorithm than our Probabilistic one. Furthermore, they did not position the performance of their probabilistic algorithm within upper and lower bounds, and did not vary the probabilities dynamically, based on observations, as our recovery function does.

III. Assumptions and Notation

We define a road network graph as follows. The n nodes in the graph $\{s_i : i = 1, 2, \ldots, n\}$, represent the intersections of the road network. The directed edges in the graph $\{d_{ij} : i, j = 1, 2, \ldots, n\}$, represent the road segments between intersections. Each edge has a cost. We take the travel time of traversing the edge as a measure of its cost. It is easy to combine the travel-time with other measures, such as walking distance between the parking block and the final destination in the parking application, but this extension is beyond the scope of this paper. Using the Dijkstra algorithm [18] and its variants, it is possible to compute the minimum cost path between any two points in the network. We use $cost_{s,d_{ij}}$ to denote the minimum cost path from a node s to the midpoint of an edge d_{ij}. Observe that for all pairs (node, edge), $cost_{s,d_{ij}}$ can be computed a priori and stored in the map.

IV. Spatio-Temporal Search Algorithms

A. Random Walk Algorithm

Positioned in an unknown environment without access to any resource-availability information, a common strategy to find a spatial resource is to search through the space randomly. We propose a *Random Walk Algorithm* that resembles this situation in a spatio-temporal search. Each time the agent reaches a node, she just randomly chooses a new direction with equal probability, and proceeds to that direction; the set of possible directions excludes the reverse, i.e. one returning to the node from which she just came. This process is repeated at the next node, and so on, until she finds a resource on some edge. This is a reasonable algorithm given that there is no information on the availability of resources, and indeed represents the way drivers currently search for parking.

B. Prophet Algorithm

We are also interested in the other end of the spectrum: how well an algorithm can perform given full information about the resource availability? To answer this question, we take it to the extreme, and assume that full information means the agent knows the exact number of available resources on each edge of the network, at each point in time, from the time when the agent starts a search. In particular, it means that the agent knows the availability on each edge at each future time point.

Observe that such an algorithm will not send an agent to the closest available edge, if it knows that the availability will be zero by the time the agent reaches the edge. In other words, such an algorithm does not simply pursue a greedy approach that pursues the closest available resource.

We develop a *Prophet Algorithm* that computes the least possible cost for the agent to find an available resource, given the "prophet" information mentioned above. The Prophet Algorithm knows for each edge d_{ij} the resource-availability a_{ij}^t at any point in time t (even if t is later than the resource-search starting time); a_{ij}^t represents the number of available resources on the edge d_{ij} at time t.

The Prophet Algorithm works as follows. For edge d_{ij}, find the minimum cost from the agent's current location to d_{ij}, and assume that the agent will travel to d_{ij} with the corresponding cost (represented by travel time). Compute a priori (i.e. before the agent starts her movement) the time when the agent arrives at d_{ij}'s midpoint, and check whether at that time there are any available resources on d_{ij}. If so, then record this minimum cost as the prophet time for d_{ij}. If not, add to this minimum cost the time that the agent has to wait at d_{ij} until there is at least one available resource on d_{ij}; and record this sum as the *prophet time* for d_{ij}. After computing the prophet times

for all edges in the network, and before starting the agent's movement, pick an edge with the minimum prophet time. This edge is the edge that the agent chooses as her destination at start-time. The prophet time for this edge is the minimum possible cost for the agent to get an available resource from her initial location. We summarize this observation as a theorem and provide a sketch of proof as follows.

Theorem. *Assume that a_{ij}^t is given for every edge and for every time point t. Let s_0 be a starting location and t_0 a starting time. Then the Prophet Algorithm has minimum cost among all algorithms that can be used by an agent that starts the resource search at s_0 and t_0.*

Sketch of proof: According to the Prophet Algorithm, for each edge d_{ij}, if it is available when the agent arrives at it using the shortest path, then the prophet time for d_{ij} is the travel time of this shortest path. This is the shortest possible time to reach d_{ij}, and therefore the time to find a resource on d_{ij} cannot be lower than this time. Similarly, for the cases where an edge d_{ij} is not available when the agent arrives at it using the shortest path, but becomes available some time after the arrival, any other path will result in a longer time to find a resource on d_{ij}. Therefore, for each individual edge, the prophet time computed by the Prophet Algorithm is the shortest possible time to find a resource on that edge. Choosing an edge whose prophet time is the smallest of all edges will guarantee that the resulting time is the shortest from s_0. ■

C. The Probabilistic Algorithm

Given probabilistic information about the availability of resources, it is possible for an agent to find a resource faster than the Random Walk Algorithm. Specifically, assume that for each edge d_{ij}, the information known to an agent is the probability p_{ij} that d_{ij} has at least one available resource.

In order to find a resource as soon as possible, two factors should be considered by the agent: the probability and the cost of an edge. A higher probability of an edge having at least one available resource means a higher likelihood that the agent will find a resource on that edge. A lower cost of an edge means that the agent can arrive at the edge faster. Therefore, consider the following formula:

$$Q_{s,d_{ij}} = \frac{p_{ij}}{cost_{s,d_{ij}}} \ .$$

Intuitively, $Q_{s,d_{ij}}$ represents the ratio between the probability and the cost of an edge; the higher the better. In the Probabilistic Algorithm, the agent could just compute the above Q value for each block, and then proceed to a block with the highest Q value. This would be an extension to probabilistic information of the following naïve, greedy, deterministic one: Go to the closest edge that has deterministic availability. In other words, this naïve deterministic algorithm is simply our Probabilistic Algorithm with all nonzero probabilities being 1.

However, in practice, we found that by merely doing this, the actual average cost for the agent to find parking is very high. This is because the probabilities are usually computed from historical or partial data. Therefore, sometimes such an edge with the highest Q value is actually not available for a long time on the actual day and time of the search. Intuitively, once the agent traverses an edge and finds no available resource, it means an immediate decrease of its probability. To capture this intuition and take advantage of observations that the agent makes during a search, we introduce the notion of *recovery functions* for probabilities.

Definition. *A recovery function for the probability of availability of an edge, $p'_{ij} = \mathrm{Recovery}(p_{ij}, t_{ij})$, is a function of time t_{ij} that is valued at zero when the agent traverses this edge and finds no available resource, and is monotonically non-decreasing until the first of the following two events occurs: 1) its initial probability p_{ij} is reached, or 2) a new traversal of that edge occurs.*

Intuitively, a recovery function discounts the probability of an edge after observing that there is no available resource on that edge, and makes it recover gradually to the prior value. Note that in order to update the recovery function, the history of the agent's movement, including times and locations, needs to be tracked.

Therefore, incorporating the recovery function, at node s the agent updates the probabilities according to her past observations, and computes $Q_{s,d_{ij}}$ for all edges in the network with p_{ij} replaced by p'_{ij}. Then, an edge with the highest $Q_{s,d_{ij}}$ would result in a higher probability and lower cost, and is chosen as the destination. However, we do not make the agent travel all the way to the edge with highest $Q_{s,d_{ij}}$. The reason is that as soon as the agent reaches the next node s' from s, the probability of the edge that agent just traversed has changed according to the recovery function. Furthermore, the cost for each edge from the new location of the agent has changed as well. Therefore, as soon as the agent has traversed an edge, we recompute $Q_{s',d_{ij}}$, and then choose a new edge to aim for. The Probabilistic Algorithm repeats this process at each node, until the agent finds an available resource on some edge.

In this paper we assume that the number of blocks that are reasonable to traverse on any search is relatively small (e.g. a parking search will not go beyond a radius of 20 blocks), and therefore the computation at each node is simply a limited breadth-first search, followed by finding the minimum in a small set of numbers. Thus, in this case, the computational complexity of the Probabilistic Algorithm is negligible.

V. Experiments

A. Simulation Environment

We conducted experiments testing the algorithms for spatio-temporal resource search on a real dataset. The dataset is from a project by San Francisco Municipal Transportation Agency (SFMTA), named SFpark [2]. In this project, in the Fishermen's Wharf area, the municipal authority buried sensors in the pavement of each parking space to detect whether or not there is a vehicle parked in that space.

The information about parking availability is published on a block level in real time. This includes the number of available

parking spaces on each block at each point in time. Specifically, the database has the schema $\langle blockId, availability, operational, timestamp\rangle$, where $timestamp$ denotes the time when a record was generated, $availability$ denotes the number of available parking spaces on a block ($blockId$), and $operational$ denotes the total number of operational parking spaces on that block. Each time the number of available parking spaces changes for any parking block, a new record is added to the database to reflect the change.

Using this database, it is possible to query the number of available parking spaces for any block at any point in time. One just needs to search the latest record for that block before the queried time; $availability$ in that record is the number of available parking spaces for the queried block at queried time.

In urban transportation systems, SFpark is a rare case where real-time and accurate information is available to spatio-temporal resource search agents. The cost of this project is prohibitive for most other municipalities. Therefore, using SFpark database as a testbed makes it possible to test probabilistic algorithms that do not assume real-time deterministic and accurate information.

To test the algorithms proposed in this paper, we used the SFpark data from Fisherman's Wharf area in San Francisco. The network in Fisherman's Wharf consists of 40 nodes and 63 edges. The size of this network is reasonable for applications like parking search. In order to find an available street parking space, one usually is only willing to search in a limited area, so that the vehicle can park not too far from the final destination.

For the Probabilistic Algorithm, we computed the probabilities of each block having at least one available parking space in the following way. For each hour in a day, we scanned the previous 20 weekdays and on each day chose a random time point during that hour. We recorded the availability of each block for those days at these random time points. And then, for each block, we computed the proportion of days on which it is available. We used this proportion as the probability of the blocks. This method resembles a process for hourly collecting historical data using a detecting vehicle with ultrasonic sensors [4]. Depending on the budget, a municipality may scan the parking blocks more or less frequently. Intuitively, more frequent scans should result in more accurate data. However, we did experiments with different lengths of time intervals ranging from 10 minutes to 3 hours, as the granularity of the probability computation, and the results did not show significant differences.

We are interested in the performance of the algorithms for different congestion levels, i.e. different levels of parking availability. As a way of increasing the congestion level, we artificially removed some proportion of parking spaces in the network. Obviously, a higher percentage of parking spaces removed means a higher level of congestion.

In our experiments, the simulations were conducted on a weekday. For each parameter combination, we repeated the simulations 10 000 times. In each simulation, the search starts at a uniformly random time between 20:00 and 21:00, and the initial location of the agent is chosen uniformly at random in the network. For comparison among the algorithms, we assumed that the agent travels in the road network at a constant speed, i.e. 5 miles per hour. Then the average time to traverse a block in this network is about 59 seconds.

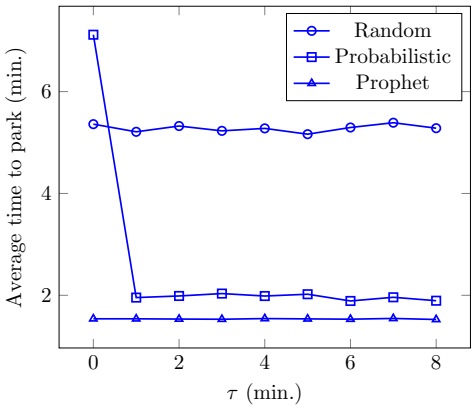

Fig. 1. Average time to park for different choices of τ in the recovery function. In this experiment, 20% of all parking spaces are removed.

B. Recovery Function for the Experiments

We use the following step function as the recovery function:

$$p'_{ij} = \text{Recovery}(p_{ij}, t_{ij}) = \begin{cases} 0, & \text{if } t_{ij} < \tau. \\ p_{ij}, & \text{otherwise.} \end{cases}$$

In this definition, the time difference between the current time and the agent's last visit of block d_{ij} is denoted as t_{ij}, and τ is used as the threshold when the probability recovers. Threshold τ is estimated from experiments, as discussed next.

C. Results

We tested the algorithms for different choices of τ in the recovery function as shown in Fig. 1. In this experiment, 20% parking spaces are removed. Note that, obviously, the Random Walk Algorithm and the Prophet Algorithm are independent of τ, because they do not use the recovery function.

For the Probabilistic Algorithm, this experiment shows the necessity for the recovery function. Note that when τ is zero, it is equivalent to the recovery function not being used. Without considering the recovery function, the Probabilistic Algorithm indeed shows a very high average time to park. From Fig. 1, it is clear that when τ is zero, the Probabilistic Algorithm performs even worse than the Random Walk Algorithm. When τ is greater than 1 minute, the Probabilistic Algorithm shows much better performance than the Random Walk Algorithm, close to that of the Prophet Algorithm.

We also compared the algorithms when different proportions of parking spaces are removed, to reflect different congestion levels. The results are shown in Fig. 2. From this figure, it is clear that the higher the congestion level is, the longer it takes on average to find parking, which confirms the intuition. Moreover, the average time to park increases much faster for the Random Walk Algorithm than the other two, as the congestion level increases. That means that for lower congestion

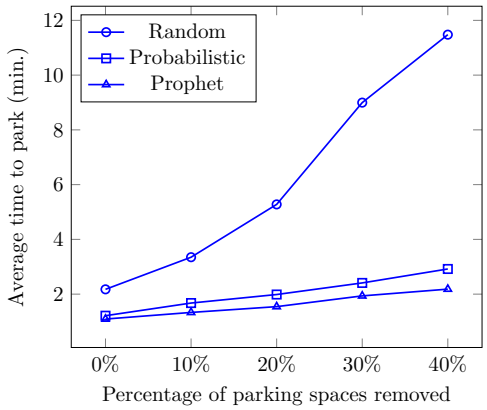

Fig. 2. Average time to park for different percentages of parking spaces removed. For the recovery function, $\tau = 4$ minutes.

levels, looking for parking without any information works reasonably well. However, as the congestion level increases, some information can reduce the search time dramatically. For example, for 30% of the spaces removed, the Random Walk Algorithm searches, on average, for over 8 minutes, whereas the Prophet and Probabilistic search on average for less than 3 minutes. Given that the average traversal time for an edge is 59 seconds, this search time of 3 minutes represents the traversal of approximately 3 blocks, instead of 8 blocks for the Random Walk Algorithm.

Another finding is that the Probabilistic Algorithm, when incorporating recovery function, works surprisingly well, showing performance that is almost as good as that of the Prophet Algorithm (the lower bound for the average cost).

VI. Conclusions

In this paper we proposed a model for spatio-temporal resource search. This framework abstracts many real-world problems, such as street parking and EV charging station search, and taxicabs searching for passengers. We considered three algorithms that assume different levels of knowledge for the agent about the availability of the resources. These are the Random Walk Algorithm that assumes no information at all, the Prophet Algorithm that assumes perfect current and future information, and the Probabilistic Algorithm that assumes partial information. For the Probabilistic Algorithm, in order to take advantage of the observations made by the agent during a search, we introduced a way of adapting probabilities dynamically via the notion of a recovery function.

To evaluate the algorithms we conducted experiments using real-world data from SFpark, with different congestion levels, and different parameter choices for the recovery function. The results, as expected, showed that the more information the agent has, the faster she is able to find a parking space. The experiments also showed that the introduction of the recovery function significantly improved the results of the searches for the Probabilistic Algorithm. And surprisingly, the performance of the Probabilistic Algorithm was almost as good as the Prophet Algorithm, which means that the historical information about the resource availability in a probabilistic form is highly valuable for the purpose of resource search.

Acknowledgments

This work was supported in part by the NSF under grants DGE-0549489, IIS-1213013 and CCF-1216096, and by the US Department of Transportation National University Rail Center (NURAIL).

References

[1] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *Proceedings of the 29th IEEE Int. Conf. on Data Engineering (ICDE)*, 2013.
[2] SFpark, http://sfpark.org/, accessed: May 2015.
[3] B. Xu, O. Wolfson, J. Yang, L. Stenneth, and P. S. Yu, "Real time street parking availability estimation," in *Proc. of 14th Intl. Conf. on Mobile Data Management (MDM)*, Milan, Italy, June 3-6, 2013.
[4] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrashekharan, W. Xue, M. Gruteser, and W. Trappe, "Parknet: Drive-by sensing of road-side parking statistics," in *MobiSys*, San Francisco, CA, June 2010.
[5] S. Ma, O. Wolfson, and B. Xu, "Updetector: Sensing parking/unparking activities using smartphones," in *Proc. of 7th Int. Workshop on Computational Transportation Science (IWCTS)*, 2014.
[6] S. Nawaz, C. Efstratiou, and C. Mascolo, "Parksense: a smartphone based sensing system for on-street parking," in *Proc. of the 19th Annual Int. Conf. on Mobile Computing and Networking (MobiCom)*. ACM, 2013, pp. 75–86.
[7] J. Boehlé, L. Rothkrantz, and M. van Wezel, "Cbprs: A city based parking and routing system," *ERIM Report Series Reference No. ERS-2008-029-LIS*, May 2008.
[8] T. Delot, S. Ilarri, S. Lecomte, and N. Cenerario, "Sharing with caution: Managing parking spaces in vehicular networks," *Mobile Information Systems*, vol. 9, pp. 69–98, 2013.
[9] Y. Geng and C. G. Cassandras, "A new "smart parking" system based on optimal resource allocation and reservations," in *Proc. of 14th Intl. Conf. on Intelligent Transportation Systems (ITSC)*, Washington, DC, USA, October 5-7 2011.
[10] D. Ayala, O. Wolfson, B. Xu, B. DasGupta, and J. Lin, "Pricing of parking for congestion reduction," in *20th Int. Conf. on Advances in Geografic Information Systems (ACM SIGSPATIAL GIS)*, Redondo Beach, CA, November 6-9 2012.
[11] ——, "Parking slot assignment games," in *Proc. of the 19th Intl. Conf. on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2011)*, Chicago, IL, November 2011.
[12] ——, "Parking in competitive settings: A gravitational approach," in *Proc. of 13th Intl. Conf. on Mobile Data Management (MDM)*, Bengaluru, India, July 23-26 2012.
[13] ——, "Spatio-temporal matching algorithms for road networks," in *20th Int. Conf. on Advances in Geografic Information Systems (ACM SIGSPATIAL GIS)*, Redondo Beach, CA, November 6-9 2012.
[14] N. J. Yuan, Y. Zheng, L. Zhang, and X. Xie, "T-finder: A recommender system for finding passengers and vacant taxis," *IEEE Transactions on Knowledge and Data Engineering*, 2013.
[15] E. Safra, Y. Kanza, N. Dolev, Y. Sagiv, and Y. Doytsher, "Computing a k-route over uncertain geographical data," in *Advances in Spatial and Temporal Databases*, ser. Lecture Notes in Computer Science, D. Papadias, D. Zhang, and G. Kollios, Eds. Springer Berlin Heidelberg, 2007, vol. 4605, pp. 276–293.
[16] V. Verroios, V. Efstathiou, and A. Delis, "Reaching available public parking spaces in urban environments using ad-hoc networking," in *IEEE Intl. Conf. on Mobile Data Management (MDM)*, 2011.
[17] G. Jossé, M. Schubert, and H.-P. Kriegel, "Probabilistic parking queries using aging functions," in *Proc. of the 21st ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*, ser. SIGSPATIAL'13. Orlando, FL, USA: ACM, 2013, pp. 442–445.
[18] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.