2016 17th IEEE International Conference on Mobile Data Management

Finding Geospatial Resources Using Uncertain Data

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

***Abstract*—In this paper, we address the problem of finding a resource in a probabilistic setting. In this problem, there are spatially located static resources and a mobile agent. The agent looks to obtain one of the resources while minimizing the cost. This cost may consist of several sub-costs the agent has to pay, from travel time to monetary cost of using a resource. We assume that the agent has no knowledge of exact availability of the resources in real-time, but some prior or partial data gives estimations of this information. This model applies to many situations that arise in urban transportation systems, such as drivers looking for street parking, or taxis looking for new customers. Our approach to the resource search problem only employs uncertain information about resource availability, minimizes the expected cost, and utilizes concepts from decision theory.**

I. Introduction

Despite the rapid growth in popularity of mobile devices, wireless embedded sensors, and location-based services, locating geographically-distributed resources still remains challenging. Certainly, with the help of smartphones, a taxi driver is able to locate potential passengers [11]; with the help of wireless embedded sensors, a vehicle is able to locate available on-street parking spaces (e.g., the SFpark project[1]). However, not all taxi passengers have smartphones and have subscriptions to taxi request services such as Uber, and even for cities like San Francisco, the cost of installing and maintaining embedded sensors for the whole city is so prohibitive that SFpark only covers a very small portion of the city.

To solve the difficulty of obtaining real-time and accurate resource availability data for resource-search, it is common to use crowdsourcing methods. In crowdsourcing, only some of the agents are used for data collection. This often results in uncertain data instead of complete and accurate data. Therefore, in this work, we deal with uncertain data.

We use a directed graph to represent a road network, and let the resources locate on the edges of the graph. In contrast to Points-of-Interest, a resource may be used by only one agent at a time. An available resource may become unavailable at a later time, and vice versa. Each edge is associated with a cost. This cost may be the cost of traversing the edge (*travel cost*, e.g. travel time), or the travel cost plus the cost of obtaining a resource on that edge (*usage cost*, e.g. walking distance from a parking block to the final destination). We call this combined cost the *general cost*. Each edge is also associated with a probability distribution of the number of available resources during a certain time period. We call any data type that represents the probability distribution an *uncertainty metric*. In this paper, we deal with two uncertainty metrics:[2] one is the probability of having at least one available resource on an edge, and the other is the mean and variance of the number of available resources on an edge.

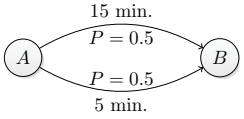

Fig. 1. An example network. The upper and lower edges have the same probability (0.5), but different travel costs (15 minutes and 5 minutes).

Using uncertain data, we address the problem of guiding the agent through the road network to efficiently find a desired resource. In other words, we aim to find an optimal search-path,[3] and to do so by an efficient algorithm.

One way of defining an optimal search-path is the Probability Maximization (PM) algorithm [9], [17]. The PM algorithm chooses a path with the maximum overall probability of finding a resource from all paths of length (cost) K or less. However, this approach has several drawbacks. First, consider the example in Fig. 1. This road network consists of two nodes and two parallel edges. Each edge has the same probability 0.5. It takes 15 minutes to traverse the upper edge, and 5 minutes to traverse the lower one. Given the maximum search time of 15 minutes, PM may choose either edge because they have the same probability. But the lower path is clearly superior because it is shorter. This difference is not captured by PM. Second, there is no straightforward way to consider usage costs in PM.

For these reasons, we focus on an alternative method, called the General Cost Minimization (GCM), to efficiently search for a resource. As the name suggests, GCM produces a path of minimum expected general cost, and does so efficiently. Consider again the example in Fig. 1. Because the shorter edge has a lower expected cost, it is preferred by GCM.

Our model applies the principle of maximum expected

[1] http://sfpark.org/

[2] The uncertain data may be obtained from historical data or partial data. One example is Xu et al. [16], where the authors used a crowdsourcing approach to approximate the parking availability per block. For this task, mobility patterns of smartphone users can be used to detect parking and unparking activities [10]. Another example is Mathur et al. [12], where the authors proposed using existing municipal vehicles equipped with GPS receivers and ultrasonic sensors to detect on-street parking availability.

[3] Observe that paths do not have to be Hamiltonian. In other words, an edge is allowed to be visited multiple times, because an unavailable edge (i.e. an edge that does not have an available resource) may later become available.

2375-0324/16 $31.00 © 2016 IEEE
DOI 10.1109/MDM.2016.23

utility (in our case, minimum expected cost) adapted from decision theory. This principle has a long history of development in the artificial intelligence community [13]. According to decision theory, if an agent's utility is specified, then an action that optimizes the expected utility will result in the most satisfactory situation. Specifically, we use a dynamic programming approach to compute the minimum expected general cost of a path within a certain length K.

Now observe that there are two variants of the resource-search problem. In one variant the probabilities are unaffected by observations. This implies that every time the agent traverses an edge e, regardless of whether the agent has traversed e previously, e will have an available resource with the same probability. However, in some applications such as parking search, since vehicles usually park for some period of time, the availability probability of a block is not independent of observations. When the agent traverses an edge e and does not find parking, it is very likely that parking will be unavailable in the next minute, even if the probability of e is high. To capture this intuition, we apply the notion of the *recovery function*, which adapts the probabilities according to observations [7]. Then we devise the Adaptive General Cost Minimization (AGCM) algorithm, which is a variant of GCM that minimizes the expected general costs for probabilities that behave according to a recovery function.

Following are the main contributions of this work:

- We provide an efficient dynamic programming algorithm that finds an optimal path with the minimum expected cost. This cost may include not only the costs of traveling the edges, but also other usage costs. Furthermore, usage costs may differ significantly between two edges, even if their travel costs are similar.
- To take advantage of observations made during a search, we adopt the notion of probability recovery function, and then devise the corresponding algorithm (AGCM) to compute a path that minimizes the expected general cost.
- We adapt our approach that uses the probability as the uncertainty metric to accommodate another uncertainty metric, i.e. the mean and variance.

The rest of the paper is organized as follows. Section II is a survey of related work. Section III presents the basic model of the resource search problem with general costs, and devises the General Cost Minimization algorithm. Section IV extends to the Adaptive General Cost Minimization algorithm by considering the recovery function. Section V shows how to use the mean and variance as another uncertainty metric. Section VI concludes the paper.

II. Related Work

As an important application of the resource search problem, the parking search problem has been tackled assuming complete and deterministic data. For example, reservation systems for parking spaces were studied by several authors [4], [5]. However, existing parking systems are not reservation-based but inherently competitive. Ayala et al. [1], [2], [3] introduced parking slot assignment games in competitive settings.

As argued, complete and deterministic information is unlikely to be available on a large scale due to the cost. Using crowdsourcing methods, resource availability information is often represented by probabilities [8], [9], [14], [15], [17].

Yuan et al. [17] propose to maximize the overall probability of finding a resource within a given time limit.

In Safra et al. [14], the probabilities are about whether a query is satisfied. For example, the query can be whether a restaurant is good according the user's preference. Therefore, the truth value of each spatial location is constant, and there is no need to revisit a location.

Jossé et al. [9] and Verroios et al. [15] assume that the exact availability of resources is given when a search request is initiated. (In contrast, we only assume probabilistic availability information.) Consequently, they solve a different problem. Then some probability decay function is used, which is a decreasing function of time for the probability that a known available resource stays available. These decay functions are similar to our recovery function, except that they go in opposite directions. Ours increases, whereas theirs decreases.

In Jossé et al. [9], Safra et al. [14], and Verroios et al. [15], variants of the traveling salesman problem are used as the theoretical solution. Since it is NP-hard, various heuristics that approximate the optimal solution are proposed by these authors. These techniques compute Hamiltonian paths for the resource search. One disadvantage of using Hamiltonian paths as the solution is that no resource location can be visited more than once. But an unavailable resource location may become available after the agent visits it for the first time.

Jossé et al. [8] considers a model similar to ours. However, the paper states that it is impossible to minimize the cost and maximize the probability of success at the same time, and proposes two solutions, i.e. minimizing the cost for the paths with probabilities greater than a threshold, or vice versa. By contrast, in this paper we show that the expected cost minimization is an effective way of combining costs and probabilities. Moreover, the exact algorithm proposed in Jossé et al. [8] performs searches by expanding all possible routes, resulting in exponential complexity. To mitigate this complexity, the authors propose some pruning techniques. By contrast, due to the application of dynamic programming, our approach is polynomial in the size of the road network.

Our previous work [6] focuses on minimizing the expected travel cost. However, the consideration of usage costs is important in practice. For example, in parking search, apart from finding parking quickly, not all parking blocks are equally preferred by the driver. Instead, the driver would prefer the blocks closer to the final destination. Despite its importance, the usage cost has not been considered in existing research [8], [9], [14], [15], [17].

Instead of probabilities, crowdsourcing methods may produce another uncertainty metric, i.e. the mean and variance of the number of available resources per location [16]. To the best of our knowledge, existing work only deals with probabilities. In this work, we propose to convert the mean and variance to the probability to accommodate them.

III. Basic Model

In this section, we formally define the problem and show how to find a path with minimum expected general cost within a certain length.

A. Problem Setup

Let us first define a *road network* which is a graph. In this graph, the vertices are the intersections of the roads, and the directed edges are the road segments connecting the intersections, indicating the allowed travel directions. We assume that there are n vertices in the road network, denoted by v_i, where $i = 1, 2, \ldots, n$. Also, let edge e_{ij} be the road segment between vertices v_i and v_j.

We define a *search-path* as a path, not necessarily simple, in the road network that is provided by any resource search algorithm to the agent. A search-path is bounded by a constant K representing the maximum number of edges in the search. In other words, K is the maximum length of a search-path.

Now we define the edge costs. Let c_{ij} denote the *cost* of e_{ij}. In general, c_{ij} consists of two types of costs. One is the *travel cost*, denoted as tc_{ij}, which is the traversal time of e_{ij}. The other is the *usage cost*, denoted as uc_{ij}, which is the cost of obtaining a resource on e_{ij}. For example, in parking search, the usage cost uc_{ij} of edge e_{ij} is the time to walk from e_{ij} to the final destination of the agent. Therefore, $c_{ij} = tc_{ij} + uc_{ij}$.

The following explains how to determine the final cost of an actual resource search. The final cost depends on whether a search is successful, i.e. whether the agent finds a desired resource following a search-path. If the search is *successful*, then the final cost is the total travel cost of the edges traversed (i.e. the prefix of the search-path), plus the usage cost of the obtained resource, if any. On the other hand, if the search is *unsuccessful*, assume that it ends at v_i, the last vertex on the search-path. Then the final cost is the total cost of the edges on the path, plus a constant β_i, This β_i denotes the additional penalty for not finding any resource after terminating the search at v_i. For example, in parking search, β_i is the cost of traveling from v_i to a private garage and parking there. In taxi-customer search, β_i is the cost for the taxi driver of returning home without finding any customer. Note that without penalty β_i, the resource search problem becomes trivial. Specifically, if $\forall i, \beta_i = 0$, the minimum expected cost of any resource search is zero and is given by a zero-length path.

It is necessary to distinguish between travel costs and usage costs. This is because the two types of costs happen at different times during a resource search, and thus affect the final cost of a search differently. Specifically, the travel cost of an edge is added to the final cost as long as the agent has traversed the edge, while the usage cost of an edge is added to the final cost only if the agent has found an available resource on it and obtained the resource. This difference means that simply adding the usage cost to the travel cost and plugging this sum to any existing algorithm will not produce correct results.

Now we define the probabilities. Denote by p_{ij} the *probability* of edge e_{ij} being available ($0 \le p_{ij} \le 1$). Intuitively, p_{ij} is the probability that e_{ij} has at least one resource available during a certain time interval (e.g. between 17:00 and 18:00). Consequently, $1 - p_{ij}$ is the probability that no resource is available on e_{ij}.

In Section III we assume that all probabilities are fixed for any given search. In Section IV we demonstrate that this assumption does not always hold in real-world applications; and we adapt the results of Section III accordingly.

B. Defining Expected General Cost

We define the expected general cost of a search-path. In our definition, the expected general cost is defined in a recursive (or stepwise) fashion. Specifically, the expected cost of a search-path is defined by the expected cost of its first edge and the expected cost of the sub-path without the first edge. This recursive definition makes it possible to devise a Bellman equation to compute a path with the minimum expected cost using dynamic programming, as demonstrated in Section III-C.

For search-path $\{v_0 \to v_1 \to \cdots \to v_K\}$, denote its expected general cost as $C_{\{v_0 \to v_1 \to \cdots \to v_K\}}$, then

$$C_{\{v_0 \to v_1 \to \cdots \to v_K\}} = \begin{cases} tc_{01} + p_{01} uc_{01} + (1 - p_{01}) C_{\{v_1 \to \cdots \to v_K\}}, & \\ & \text{if } uc_{01} \le C_{\{v_1 \to \cdots \to v_K\}}. \\ tc_{01} + C_{\{v_1 \to \cdots \to v_K\}}, & \text{otherwise.} \end{cases} \tag{1}$$

The first case in Equation (1) deals with the situation when the usage cost of the first edge e_{01} is not greater than the expected cost of the rest of the path. In this case, after traversing the first edge (the first term tc_{01}), the second term ($p_{01} uc_{01}$) gives the expected cost of finding the resource on the first edge of the path, and ending the search there; and the third term ($(1 - p_{01}) C_{\{v_1 \to \cdots \to v_K\}}$) is the expected cost of not finding a resource and continuing the search on the rest of the path. The second case in Equation (1) is when the usage cost of the first edge e_{01} is greater than the expected cost of the rest of the path. In this case, regardless of whether there exists an available resource on the first edge, the agent continues the search without obtaining any resource on e_{01}. Similarly:

$$C_{\{v_1 \to \cdots \to v_K\}} = \begin{cases} tc_{12} + p_{12} uc_{12} + (1 - p_{12}) C_{\{v_2 \to \cdots \to v_K\}}, & \\ & \text{if } uc_{12} \le C_{\{v_2 \to \cdots \to v_K\}}. \\ tc_{12} + C_{\{v_2 \to \cdots \to v_K\}}, & \text{otherwise.} \end{cases}$$

$$\cdots$$

$$C_{\{v_{K-1} \to v_K\}} = \begin{cases} tc_{K-1,K} + p_{K-1,K}\ uc_{K-1,K} + (1 - p_{K-1,K}) \beta_K, & \\ & \text{if } uc_{K-1,K} \le \beta_K. \\ tc_{K-1,K} + \beta_K, & \text{otherwise.} \end{cases}$$

C. Expected General Cost Minimization

In this subsection, we show how to compute the minimum expected general cost of all paths within length K using dynamic programming. The result is composed of two parts: an optimal path and an auxiliary action indicator sequence.

Recall that with usage costs, obtaining the first-found available resource along the optimal path may not be the optimal action. To achieve the minimum expected general cost, it is crucial to make sure to obtain an available resource only if its usage cost is lower than the expected general cost of continuing the search along the remaining optimal path. To do this, we produce an action indicator sequence during the computation. This sequence specifies whether the agent should obtain a resource if one is found available along the optimal path.

Before presenting the detailed computation, we first define some notation. For $k \in \{1, \ldots, K\}$, let the *k-step look-ahead from node* v_i be the set of all paths starting from v_i with length k or less. Denote by C_i^k the minimum expected general cost of a path in the k-step look-ahead from v_i. Assume that v_j is an immediate successor of v_i in the network. Denote by C_{ij}^k the expected general cost of a path in the k-step look-ahead from v_i, for which: 1) the first two vertices are v_i and v_j, and 2) the expected general cost of the remaining sub-path within length $k-1$ is the minimum among all paths within length $k-1$ from v_j. We call the pair (v_i, k) a *decision point*.

The following equations indicate how to compute C_i^k and C_{ij}^k recursively, and produce the optimal path and the action indicator sequence:[4]

$$C_i^k = \begin{cases} \min\limits_{\forall j, \text{ s.t. } e_{ij} \text{ exists}} \left\{C_{ij}^k, \beta_i\right\}, & \text{if } k > 0. \\ \beta_i, & \text{if } k = 0. \end{cases} \tag{2}$$

$$C_{ij}^k = \begin{cases} tc_{ij} + p_{ij} uc_{ij} + (1 - p_{ij}) C_j^{k-1}, & \text{if } uc_{ij} \le C_j^{k-1}. \\ tc_{ij} + C_j^{k-1}, & \text{otherwise.} \end{cases} \tag{3}$$

$$next_i^k = \begin{cases} \arg\min\limits_{\forall j, \text{ s.t. } e_{ij} \text{ exists}} \left\{C_{ij}^k\right\}, & \text{if } C_i^k \neq \beta_i. \\ -1, & \text{if } C_i^k = \beta_i. \end{cases} \tag{4}$$

$$skip_i^k = \begin{cases} \text{false}, & \text{if } j_0 = next_i^k \neq -1 \text{ and } uc_{ij_0} \le C_{j_0}^{k-1}. \\ \text{true}, & \text{if } j_0 = next_i^k \neq -1 \text{ and } uc_{ij_0} > C_{j_0}^{k-1}. \end{cases} \tag{5}$$

Equations (2) and (3) compute C_i^k and C_{ij}^k from C_j^{k-1}. Equation (3) is based on Equation (1). Equations (4) and (5) are used to produce the optimal path and the action indicator sequence as follows. Assume that v_0 is the starting node, then $l_1 = next_0^K, l_2 = next_{l_1}^{K-1}, \ldots$, until the first k_0 such that $next_{l_{k_0}}^{K-k_0} = -1$. The resulting path $\{v_0 \to v_{l_1} \to \cdots \to v_{l_{k_0}}\}$ is the optimal path that always minimizes future expected general costs as the agent traverses along this path. In Equation (5), $skip_i^k$ is a Boolean value that assists the agent when conducting the resource search. Specifically, at decision point (v_i, k), if $skip_i^k$ = true, then the agent should proceed to node $j = next_i^k$ and continue the resource search, even if edge e_{ij} has an available resource. This is because $skip_i^k$ being true means that the usage cost of e_{ij} is greater than the expected general cost of the rest of the optimal path. On the other hand, if $skip_i^k$ = false, then the agent should obtain a resource from e_{ij} if there is one available.

[4]"s.t." is short for "such that".

Algorithm 1 General Cost Minimization Algorithm (GCM)

Input: Network $\langle E, V\rangle$, $\{tc_{ij}\}$, $\{uc_{ij}\}$, $\{\beta_i\}$, $\{p_{ij}\}$
Output: $\{C_i^k\}$, $\{next_i^k\}$, $\{skip_i^k\}$

for $i = 1, 2, \ldots, n$ **do**
  $C_i^0 \leftarrow \beta_i$
end for
for $k = 1, 2, \ldots, K$ **do**
  for $i = 1, 2, \ldots, n$ **do**
    $C_i^k \leftarrow \infty$
    for all v_j s.t. e_{ij} exists **do**
      if $uc_{ij} \le C_j^{k-1}$ **then**
        $C_{ij}^k \leftarrow tc_{ij} + p_{ij} uc_{ij} + (1 - p_{ij}) C_j^{k-1}$
        $skip \leftarrow$ false
      else
        $C_{ij}^k \leftarrow tc_{ij} + C_j^{k-1}$
        $skip \leftarrow$ true
      end if
      if $C_{ij}^k < C_i^k$ **then**
        $C_i^k \leftarrow C_{ij}^k$
        $next_i^k \leftarrow j$
        $skip_i^k \leftarrow skip$
      end if
    end for
    if $\beta_i < C_i^k$ **then**
      $C_i^k \leftarrow \beta_i$
      $next_i^k \leftarrow -1$
    end if
  end for
end for

The following confirms the optimality of this approach:

Theorem. *The expected cost C_i^K computed by Equations (2) and (3) is the minimum among all paths with length K or less, starting at vertex v_i. Furthermore, C_i^K can be computed in time $O(ndK)$, where d is the maximum degree of a node.*

Proof Sketch: This can be proven by induction on K. The proof considers all possible relationships between the usage cost of any edge e and the expected cost of any path starting from e. ■

Equations (2) and (3) can be regarded as the Bellman equation [13] in a dynamic programming method for computing the minimum expected cost of any path with length K or less. This is because each C_i^k only needs to be computed once.

Algorithm 1 (GCM) computes the minimum expected general costs and the corresponding optimal paths and action indicator sequences. Overall, it builds the minimum expected costs bottom-up by Equations (2) and (3) using dynamic programming for increasing k, until the K is reached. Clearly, Algorithm 1 has a time complexity of $O(ndK)$.

IV. Adaptive Probabilities

In this section, we present a modified version of the basic resource search model that allows adaptive probabilities. Specifically, in the model of Section III, the probabilities are assumed to be constant for a given time interval. This implies that every time the agent traverses an edge e, regardless of whether the agent traversed e previously, e will have an available resource with the same probability. This assumption represents real-world situations in some applications. However, in applications such as the parking search, since vehicles usually park for some period of time, the availability probability of a block is not independent of observations. When the agent traverses a parking block and does not find parking, it is very likely that parking will be unavailable in the next minute, even if the probability of this block is high. The model in this section is adapted to take observations into consideration: the observation that there is no availability at the moment is a good indicator that this block will have a lower probability of availability for some time in the future. Therefore, in this section, we allow the probability of an edge to decrease after observing that it is unavailable.

A. Probability Recovery Function

In order to utilize the observations during a search, we define the *recovery function* for the probabilities as follows [7]:

Definition. *A* recovery function $p'_{ij} = \text{Recovery}(p_{ij}, t_{ij})$ *for the probability of edge e_{ij}, where t_{ij} is the time that has elapsed since edge e_{ij}'s last traversal, is a function of time that is valued at zero when the agent traverses e_{ij} and finds no available resource, and monotonically increases until either of these events occurs: 1) the prior probability p_{ij} is reached, or 2) a new traversal of e_{ij} occurs. In the first case the probability stays at p_{ij}, and in the second it drops back to zero.*

B. Expected Cost Minimization with Recovery Function

In this subsection, we describe a resource search algorithm, derived from GCM, that incorporates the probability recovery function. We call this modified algorithm the *Adaptive General Cost Minimization* (AGCM) algorithm.

Now we make an important assumption on the probability recovery function that we use for AGCM. We assume that the recovery function recovers a probability to its prior value within in a limited time t_0. That is, $\forall t_{ij} \geq t_0, p'_{ij} = \text{Recovery}(p_{ij}, t_{ij}) = p_{ij}$. With this restriction, only a limited history of visited edges needs to be tracked to compute the recovered probabilities. In other words, it prevents the need of tracking a complete traversal history. Denote by h the minimum number of edges needed to be tracked, then $h = \lceil t_0/c_{min} \rceil$, where $c_{min} = \min_{\text{all } e_{ij}} c_{ij}$.

Now we show how to find the minimum expected general cost of paths with length at most K considering the recovery function. In order to keep track of a history of the last h edges that the agent has considered, we expand the concept of decision point so it contains a history of previously traversed nodes. Specifically, decision point $(v_i|\{v_{i-h}, \ldots, v_{i-1}\}, k)$ means that the agent is currently at node v_i in k-step look-ahead, and that $\{v_{i-h}, \ldots, v_{i-1}\}$ is the previous h nodes the agent has traversed before v_i. Similarly to the basic model, denote by $C^k_{i|i-h,\ldots,i-1}$ the minimum expected general cost for decision point $(v_i|\{v_{i-h}, \ldots, v_{i-1}\}, k)$. Denote by $C^k_{i,j|i-h,\ldots,i-1}$ the expected general cost of the paths for decision point $(v_i|\{v_{i-h}, \ldots, v_{i-1}\}, k)$, such that 1) v_i and v_j are the first two nodes of the path, and 2) the remaining sub-path starting from v_j that is consistent with the history, has a minimum expected general cost with $(k-1)$-step look-ahead. Then similar to Equations (2), (3), (4), and (5), these two costs can be computed as:

$$C^k_{i|i-h,\ldots,i-1} = \begin{cases} \min\limits_{\forall j, \text{ s.t. } e_{ij} \text{ exists}} \left\{ C^k_{i,j|i-h,\ldots,i-1}, \beta_i \right\}, & \\ & \text{if } k > 0. \\ \beta_i, & \text{if } k = 0. \end{cases} \tag{6}$$

$$C^k_{i,j|i-h,\ldots,i-1} = \begin{cases} tc_{ij} + p'_{ij} uc_{ij} + (1 - p'_{ij}) C^{k-1}_{j|i-h+1,\ldots,i}, & \\ & \text{if } uc_{ij} \leq C^{k-1}_{j|i-h+1,\ldots,i}. \\ tc_{ij} + C^{k-1}_{j|i-h+1,\ldots,i}, & \text{otherwise.} \end{cases} \tag{7}$$

$$p'_{ij} = \text{Recovery}(p_{ij}, t_{ij}). \tag{8}$$

$$next^k_{i|i-h,\ldots,i-1} = \begin{cases} \arg\min\limits_{\forall j, \text{ s.t. } e_{ij} \text{ exists}} \left\{ C^k_{i,j|i-h,\ldots,i-1} \right\}, & \\ & \text{if } C^k_{i|i-h,\ldots,i-1} \neq \beta_i. \\ -1, & \text{if } C^k_{i|i-h,\ldots,i-1} = \beta_i. \end{cases} \tag{9}$$

$$skip^k_{i|i-h,\ldots,i-1} = \begin{cases} \text{false}, & \text{if } j_0 = next^k_{i|i-h,\ldots,i-1} \neq -1 \\ & \text{and } uc_{ij_0} \leq C^{k-1}_{j_0|i-h+1,\ldots,i}. \\ \text{true}, & \text{if } j_0 = next^k_{i|i-h,\ldots,i-1} \neq -1 \\ & \text{and } uc_{ij_0} > C^{k-1}_{j_0|i-h+1,\ldots,i}. \end{cases} \tag{10}$$

Note that in the beginning of a search, the length of history may be smaller than h edges. In this case, h in above equations should be replaced by the actual length of history.

Similarly to the basic model in Section III, based on these equations, a dynamic programming algorithm (i.e. AGCM) can be developed. Specifically, $C^k_{i|i-h,\ldots,i-1}$ is computed by $C^{k-1}_{j|i-h+1,\ldots,i-1,i}$ of every successor node v_j of v_i. This requires the computation of the minimum expected cost for each decision point. For node v_i, there are at most d^h possible predecessor paths, where d is the maximum degree of a node in the graph. Therefore, for a (v_i, k) pair, there are at most d^h decision points.

Considering the total number of decision points, the time complexity of AGCM is $O(nd^{h+1}K)$. If the length bound of the recovery function h is constant, this time complexity is polynomial in the size (n nodes) of the graph. Although it is exponential in h, we found by experimentation that only a

short history for the recovery function is needed to achieve satisfactory performance for real-life problems. In addition, as in the basic model, given the prior probabilities, these optimal paths can be precomputed for each possible starting location and stored for later use. Therefore, for a reasonable h, AGCM is scalable in terms of the network size.

V. Mean and Variance as Uncertainty Metric

The uncertainty metric obtained from crowdsourcing is often the mean and variance of the number of available resources per edge, instead of probability [16]. In this section, we present how to convert means and variances to probabilities so our approach can apply to them.

To convert means and variances to probabilities, we assume that the number of available resources on an edge during a certain time period follows a discretized Gaussian distribution. Specifically, assume that for an edge e, the number of available resources on e is R ($R \in \{0, \ldots, R_{max}\}$), where R_{max} is the maximum capacity of e. We assume that R is associated with a real-valued random variable X that follows Gaussian distribution. Then, $\Pr(R = 0) = \Pr(X < 0.5), \Pr(R = 1) = \Pr(0.5 \leq X < 1.5), \ldots, \Pr(R = R_{max} - 1) = \Pr(R_{max} - 1.5 \leq X < R_{max} - 0.5), \Pr(R = R_{max}) = \Pr(X \geq R_{max} - 0.5)$. Therefore, the probability of edge e is $\hat{p} = \Pr(R \geq 1) = \Pr(X \geq 0.5) = \int_{0.5}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx$, where μ is the mean and σ^2 is the variance of the number of available resources. Assume that we are given m samples of the number of available resources on e, denoted by $r_1, \ldots, r_m$. Then the mean μ can be estimated by sample mean $\hat{\mu} = \bar{r} = \frac{1}{m}\sum_{l=1}^{m} r_l$; the variance σ^2 can be estimated by unbiased sample variance $\hat{\sigma}^2 = s^2 = \frac{1}{m-1}\sum_{l=1}^{m}(r_l - \bar{r})^2$. Replacing the probabilities in GCM and AGCM with above $\hat{p}$, one can use the mean and variance of the number of available resources for each edge to compute the optimal paths.

VI. Conclusions

In this paper, we use the minimum expected general cost as the optimization criterion for the spatio-temporal resource search problem with uncertain data. Clearly, it is beneficial to consider general costs that include usage costs in the resource search problem. We show that our optimization criterion is advantageous over other possible optimization criteria such as Probability Maximization.

In applications such as on-street parking search, traversing a resource location and not finding the desired resource means an immediate decrease of its probability. To utilize this observation, we propose the Adaptive General Cost Minimization algorithm. This algorithm has a time complexity that is exponential in the length of the path that needs to be tracked during recovery. However, its efficiency is still reasonable because the length of recovery-path is usually very small, according to our experiments. For example, for SFpark data the probability of a block recovers to its initial value within 2 minutes, thus the algorithm is exponential in the number of edges that can be traversed in 2 minutes.

For some data collection methods, the uncertain data about the availability of the resources may be the mean and variance of the number of resources for each location instead of the probability of having a least one available resource. We use discretized Gaussian distribution to convert the mean and variance to the probability.

Acknowledgments

This work was supported in part by the NSF under grants IIS-1213013 and IIP-1534138.

References

[1] D. Ayala, O. Wolfson, B. Xu, B. DasGupta, and J. Lin. 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.

[2] D. Ayala, O. Wolfson, B. Xu, B. DasGupta, and J. Lin. Parking in competitive settings: A gravitational approach. In *Proc. of 13th Intl. Conf. on Mobile Data Management (MDM)*, Bengaluru, India, July 23-26 2012.

[3] D. Ayala, O. Wolfson, B. Xu, B. DasGupta, and J. Lin. 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.

[4] 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.

[5] T. Delot, S. Ilarri, S. Lecomte, and N. Cenerario. Sharing with caution: Managing parking spaces in vehicular networks. *Mobile Information Systems*, 9:69–98, 2013.

[6] Q. Guo and O. Wolfson. Presents: Probabilistic resource-search networks. In *Proceedings of the 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15. ACM, 2015.

[7] Q. Guo, O. Wolfson, and D. Ayala. A framework on spatio-temporal resource search. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2015 11th International*, August 2015.

[8] G. Jossé, K. A. Schmid, and M. Schubert. Probabilistic resource route queries with reappearance. In *Proceedings of the 18th International Conference on Extending Database Technology*, EDBT '15, pages 445–456, 2015.

[9] 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*, SIGSPATIAL'13, pages 442–445, Orlando, FL, USA, 2013. ACM.

[10] 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.

[11] 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.

[12] 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.

[13] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (Second Edition)*. Prentice Hall, 2003.

[14] E. Safra, Y. Kanza, N. Dolev, Y. Sagiv, and Y. Doytsher. Computing a k-route over uncertain geographical data. In D. Papadias, D. Zhang, and G. Kollios, editors, *Advances in Spatial and Temporal Databases*, volume 4605 of *Lecture Notes in Computer Science*, pages 276–293. Springer Berlin Heidelberg, 2007.

[15] 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.

[16] 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.

[17] 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.