Route Search over Probabilistic Geospatial Data

Yaron Kanza[1,*], Eliyahu Safra[2], and Yehoshua Sagiv[3,**]

[1] Technion—Israel Institute of Technology
kanza@cs.technion.ac.il
[2] Environmental Systems Research Institute
esafra@esri.com
[3] The Hebrew University of Jerusalem
sagiv@cs.huji.ac.il

Abstract. In a *route search* over geospatial data, a user provides terms for specifying types of geographical entities that she wishes to visit. The goal is to find a route that *(1)* starts at a given location, *(2)* ends at a given location, and *(3)* travels via geospatial entities that are relevant to the provided search terms. Earlier work studied the problem of finding a route that is *effective* in the sense that its length does not exceed a given limit, the relevancy of the objects is as high as possible, and the route visits a single object from each specified type. This paper investigates route search over *probabilistic geospatial data*. It is shown that the notion of an effective route requires a new definition and, specifically, two alternative semantics are proposed. Computing an effective route is more complicated, compared to the non-probabilistic case, and hence necessitates new algorithms. Heuristic methods for computing an effective route, under either one of the two semantics, are developed. (Note that the problem is NP-hard.) These methods are compared analytically and experimentally. In particular, experiments on both synthetic and real-world data illustrate the efficiency and effectiveness of these methods in computing a route under the two semantics.

1 Introduction

The recent growth of the world-wide web has made geographical applications prevalent and accessible to many different users. This has raised the need for geographical applications that are adapted to the environment of the web in the following three aspects. First, applications on the web should be simple, in order to be suitable for novice users. Secondly, in many cases, the applications should be able to deal with heterogeneous data, and in particular with data that is inaccurate and incomplete. Thirdly, applications should be highly efficient in order to be provided as web services. Efficiency is essential also when considering

[*] The work of this author was supported by the German-Israeli Foundation for Scientific Research & Development (Grant 2165-1738.9/07) and by a grant from the Goldstein UAV and Satellite Center at the Technion.

[**] The work of this author was supported by the German-Israeli Foundation for Scientific Research & Development (Grant 973-150.6/2007).

geographical applications that should run on devices with a limited computation power, such as cellular phones or car navigation systems.

One of the geographic applications that is recently being adapted to the web is a *route search*. In a route search, the goal is to find a route that starts at a given location, ends at a given location and travels via geographical entities that satisfy the search terms. The search terms define which geographical entities the route should visit in order to satisfy the needs of the user—*i.e.*, they define the types of these entities and additional constraints about them. For example, suppose that Alice has to go from her office to a meeting at a certain place, and on her way she wants to stop by at an ATM, a restaurant that serves a specific kind of food and a pharmacy. A route-search application should receive these requirements as a query and then return a route that starts at the office of Alice, ends at the location of the meeting, and travels via an ATM, a restaurant and a pharmacy, not necessarily in this specific order.

Data on the web might be inaccurate and incomplete. This can be caused by changes in the real world that have not yet been updated in the database. For instance, a restaurant has been closed, but still appears in the database. It can also be the result of errors caused by incorrect integration of data that arrive from different sources on the web. An integration allows utilizing different sources of information; however, an incorrect join leads to an inaccurate result. For instance, if one restaurant is erroneously joined with the menu of another restaurant, query results over such data may not be correct.

Uncertainty is affected not just by the data, but also by the accuracy of the query. Commonly, queries on the web are simple and imprecise, which leads to uncertainty regarding whether a given geographical entity satisfies the user requirements. For example, when Alice arrives at the restaurant, she might find that it does not serve food that complies with her diet. Similarly, the pharmacy may be closed or may not include the medicine she needs.

For coping with inaccuracy and incompleteness, we use a probabilistic model. In a probabilistic model, each geographic object is assigned a *probability-of-success*. The probability-of-success indicates how likely it is that the geographical entity represented by the object will satisfy the user specifications. For example, when a user asks for a route that travels via a place that sells pizza, a place called "Pizza Hut" will receive a very high probability of success. Similarly, a place called "Italian Restaurant" will also receive a high probability of success. However, for a place called "Japanese Restaurant," the assigned probability should be much lower. This assigned probability can be based on collected statistics, on information-retrieval techniques and on user profiling, *e.g.*, for a user whose profile indicates he has a low income, the probability-of-success of expensive restaurants should be reduced in comparison to the probability assigned to economy restaurants. How exactly to assign these statistics is, however, beyond the scope of this paper.

When computing a route over probabilistic data, two approaches are possible. In the first approach, the route should travel via exactly one object of each required type, while attempting to go via the objects whose probability-of-success is as high as possible; *e.g.,* in the above example, the generated route will go via exactly one restaurant, one ATM and one pharmacy. This approach is problematic in the following sense. In this approach, the probability-of-success for a route is the probability that all the objects on this route will satisfy the user. Yet, if for one particular type of entities, all the objects of that type have low probabilities, then the probability of the entire route will also be low. For instance, if in the route search of Alice, all the restaurant objects in the database have a low probability, the system will not be able to create a route that has a high probability of success.

In the second approach, the route can go via several objects of each type and the probability-of-success is the probability that for each type, the route has at least one object of that type that meets the user requirements. For instance, a route for Alice may travel via several ATMs to increase the probability that Alice will visit a functioning ATM. In this paper, we investigate only this approach. Note that in order for the route to be effective, it should not be too long. Thus, the set of objects that the route travels through should be chosen carefully.

In this paper, we propose two semantics for route-search queries over probabilistic data. Under the *bounded-length semantics*, a distance limit ℓ is provided and the goal is to find the route with the highest probability of success among the routes that satisfy the query and whose length does not exceed ℓ. Under the *bounded-probability semantics*, a probability value p is given, and the goal is to find the shortest route among the routes that satisfy the query and provide probability of success not smaller than p.

The bounded-length semantics is appropriate when the length of the route must not exceed a given value. For example, a customer of a rental-car agency would like not to exceed the traveling distance that is included in the base price. The bounded-probability semantics is suitable when the user can be flexible with the length, yet would like to guarantee a certain probability of success.

As a web application, a route-search algorithm should be highly efficient. However, computing a route under either the bounded-length semantics or the bounded-probability semantics is an NP-hard problem—simple reductions from the traveling-salesman problem (TSP) show that. Hence, in this paper, we present heuristics for the problems.

The paper is organized as follows. In Section 2, we present the formal framework and define the two semantics of route search over probabilistic data. In Section 3, we survey recent papers on route search, and we compare the computation of a route under the two semantics to similar problems that exist in the literature. Heuristic algorithms for computing a route under the two semantics are presented in Section 4. In Section 5, we present the results of experiments that illustrate the effectiveness and efficiency of our algorithms. Finally, we conclude in Section 6.

2 Probabilistic Route Search

In this section, we present our formal framework and define the concept of a *probabilistic route search*.

2.1 Geographical Datasets

A *geo-spatial dataset* is a collection of geo-spatial objects. Each object represents a real-world geographical entity and has a location—the location of an object is the location of the entity it represents. An object may have additional spatial and non-spatial attributes. Height and shape are examples of spatial attributes. Address and name are examples of non-spatial attributes. We assume that locations are points and are unique, *i.e.*, different objects have different locations. For objects that are represented by a polygonal shape and do not have a specified point location, we consider the center of mass of the polygonal shape to be the point location.

For simplicity, we measure the distance between two objects in terms of the Euclidean distance between their point locations. However, our algorithms do not assume that distances are Euclidean and, hence, they are applicable also when movement is constrained to a road network. We denote the distance between two objects o_1 and o_2 by $distance(o_1, o_2)$. Similarly, if o is an object and l is a location, then $distance(o, l)$ is the distance from o to l.

2.2 Search Queries

Users specify what entities they would like to visit by *search queries.* Formally, a query is a pair $Q = (W, C)$, where *(1)* W is a set of keywords, *and (2)* C is a set of constraints having the form $A \diamond v$, such that A is an attribute name, v is a value and $\diamond$ is a comparison symbol among $=, <, >, \neq, \leq$ and $\geq$. For instance, `Hotel, Wireless Internet Access, rank` ≥ 3`, price` ≤ 100 specify that the user would like to go via a hotel that provides an Internet wireless connection, has a ranking of at least three stars and a rate that does not exceed $100.

In a non-probabilistic setting, there is a clear-cut notion of when an object o *satisfies* a given query. In our framework, each object has some degree of *relevancy* to a given search query Q. That degree is stated as a *probability-of-success* (or *probability*, for short), which is a value between 0 and 1. This probability indicates how high are the chances that o satisfies the user requirements. We denote the probability of an object o by $Pr(o)$.

Methods for determining the probability of an object are beyond the scope of this paper. In a nutshell, initial probabilities are determined when objects are created or updated in the database. Those probabilities may be derived by estimating the reliability of the information sources, as well as other factors. In addition, rules could be applied in order to adjust the probability of an object according to the query at hand. For example, if an attribute of a restaurant object contains the string "Italian food," then a rule may determine that the restaurant serves pizza with probability 0.9. Finally, the underlying model determines how to compute the probability of joint events. For example, the probability that a restaurant object serves pizza and charges moderate prices could be computed by assuming that these two events are independent. In summary, we assume that the probability of each object is given and incorporates all the relevant factors.

2.3 Route-Search Queries

In a *route-search query*, the user specifies a *source location*, a *target location* and the entities that the route should visit. We represent a route-search query as a triplet $R = (s, t, \mathcal{Q})$, where s is a source location, t is a target location and $\mathcal{Q}$ is a set of search queries.

Example 1. Consider again the route-search task of Alice presented in Section 1. A suitable route-search query for this task should include *(1)* the location s of the office of Alice, *(2)* the location t where the meeting should be held, and *(3)* the following three search queries: $Q_1 = \{$`restaurant esoteric food`$\}$, where "esoteric food" refers to the type of diet Alice needs; $Q_2 = \{$`ATM`$\}$; and $Q_3 = \{$`pharmacy`$\}$.

Consider a route-search query $R = (s, t, \mathcal{Q})$, and let $Q_1, \ldots, Q_m$ be the search queries of $\mathcal{Q}$. The *result* of Q_i, denoted by A_i, comprises the objects of the database that are relevant to Q_i. We assume that the A_j are pairwise disjoint. In other words, distinct search queries of $\mathcal{Q}$ refer to different types of objects. For example, one search query is about hotels, another is concerning restaurants, etc. A *pre-answer* to R is a route that starts at s, ends at t and goes via objects of the results $A_1, \ldots, A_m$. That is, a route is a sequence $s, o_1, \ldots, o_k, t$, such that each o_i belongs to some A_j. The *length* of the route is the sum of the distances between consecutive objects, that is,

$$distance(s, o_1) + \Sigma_{i=1}^{k-1} distance(o_i, o_{i+1}) + distance(o_k, t).$$

As mentioned earlier, each object o_i on the route has a probability $Pr(o_i)$ that indicates its relevancy to the corresponding search query. Let ρ denote the above route, namely, $s, o_1, \ldots, o_k, t$. The *probability-of-success* of ρ (or *probability*, for short), denoted by $Pr(\rho)$, is given by the following equation.

$$Pr(\rho) = \Pi_{j=1}^{m}(1 - \Pi_{o_i \in A_j}(1 - Pr(o_i))).$$

This equation is derived as follows. First, $1 - Pr(o_i)$ is the probability that object $o_i \in A_j$ does not satisfy the requirements implied by the search query Q_j. So, for a given A_j, the product $\Pi_{o_i \in A_j}(1 - Pr(o_i))$ is the probability that A_j has no relevant object on the route ρ. Hence, $1 - \Pi_{o_i \in A_j}(1 - Pr(o_i))$ is the probability of the complement event, namely, A_j has at least one relevant object o_i on the route ρ. The final product over all the A_j is the probability of the route ρ.

The *answer* to a route-search query is a pre-answer chosen according to a specific semantics. In this paper, we present two semantics for route-search queries over probabilistic data.

Bounded-Length Semantics. Let ℓ be a given distance limit. Under the *bounded-length semantics*, the answer is the pre-answer that has the highest probability of success among the pre-answers whose length does not exceed ℓ.

Bounded-Probability Semantics. Let p be a given probability threshold. Under the *bounded-probability semantics*, the answer is the pre-answer that has the shortest length among the pre-answers whose probability of success is not smaller than p.

3 Related Work

Some variations of route search have been investigated in earlier works. Safra et al. [13] studied the case of a route search over uncertain data where all the objects belong to the same set. This is similar, but not identical, to a special case of the route search presented in this paper, where there is a single search query in R and the bounded length semantics is used. This special case also has some similarity to the *orienteering problem*. In the orienteering problem, the input consists of a distance limit, a start location and a set of objects where each object has a score. The problem is to compute a route that (1) starts at the given starting location, (2) have a length that does not exceed the given distance limit and (3) goes via objects whose total score is maximal. The orienteering problem has been studied extensively [2,5] and several heuristics [1,4,8,9,16] and approximation algorithms [12] were proposed for it.

There are two main differences between orienteering and the problem of computing a route under the bounded-length semantics. First, in a route search, the objects are divided into sets (the sets are the answers to the search queries) and an object from each set must be visited in order to have a probability-of-success greater than zero. In the orienteering problem, objects differ only in their location and score.

Secondly, in the orienteering problem, when an object is added to a route, its effect on the total score is always equal to its score, while in a probabilistic route search, the effect of an object on the probability depends on both the probability of the object and the probabilities of objects in the route. For instance, an object with score 0.5 will always add 0.5 to the score of an orienteering sequence. However, in a probabilistic route search, an object of set A with probability 0.5 will have a different effect in each of the following three cases: a route that does not contain any object of A, a route that contains an object of A with probability 0.4, and a route that contains an object of A with probability 1.

Because of these differences, there is no simple way of using heuristic or approximation algorithms of the orienteering problem to solve a route search, even when the route search comprises a single search query.

A route search over objects that are partitioned into sets has been studied by Kanza et al. [7]. However, they considered a semantic of route search that is different from the semantics suggested in this paper. First, under their semantics, the route must visit exactly one object of each set while in the probabilistic case, a route can visit several objects of each set. Secondly, as explained above, in the non-probabilistic case, the effect of an object on a route depends on the object, whereas in the probabilistic case, it depends on both the object and the route. Therefore, algorithms and heuristics for the non-probabilistic case do not solve a route search over probabilistic data. For instance, consider a search for a phone booth and the following two routes, having the same length. A route that goes via a single place which has a phone booth with probability 0.6, and a route that goes via 10 places where a phone booth can exist with probability 0.5. A non-probabilistic search will prefer the first route while the probability-of-success is much greater in the second route.

Several papers [3,10,11,14] study route-search queries over datasets in which objects have neither scores nor probabilities. The work of [15] investigates two variants of the shortest-route problem. In one, there should be exactly k intermediate points. In the second, the distance between any two consecutive points should not exceed a given bound. In the framework of [15], there are no scores, probabilities, or different types of points. In [6], they consider a user who travels a predetermined route in a road network. Their goal is to modify the route so that a new point is visited in a manner that optimizes some spatial preferences.

4 Algorithms

Under the bounded-length semantics, even a restricted version of the problem is NP-hard. To prove it, consider an instance of the following decision problem: does the traveling-salesman route has a length of at most ℓ. We construct a database D and a route-search query $R = (s, t, \mathcal{Q})$ as follows. The objects of D are those of the traveling-salesman problem. An arbitrary object o of D is chosen as both s and t. $\mathcal{Q}$ has a single search query Q, such that every object of $D \setminus \{o\}$ is relevant to Q with probability 0.5. The distance bound is ℓ. Clearly, the length of the traveling-salesman route is at most ℓ if and only if the answer to the route-search query (under the bounded-length semantics) has probability $1 - 0.5^{n-1}$, where n is the number of objects in D. Under the bounded-probability semantics, we can use the same reduction in order to compute the length of the traveling-salesman route. For that, we have to choose $1 - 0.5^{n-1}$ as the probability threshold.

4.1 Heuristics for the Bounded-Length Semantics

Since route search is a hard problem, we present heuristics instead of algorithms that provide exact answers. In this section, we give four heuristics for route search under the bounded-length semantics. These heuristics were devised to be efficient, which is a necessary requirement when developing web applications.

Throughout this section, $R = (s, t, \mathcal{Q})$ is the given route-search query, where s is the start location, t is the target location, and $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ are the search queries. D is the dataset over which R is posed and $A_1, \ldots, A_m$ are the results of the search queries, *i.e.,* $A_i = Q_i(D)$ for all $1 \leq i \leq m$. The probability-of-success of each object $o \in D$ is denoted by $Pr(o)$. Finally, ℓ is the length limit.

Ratio-Greedy Heuristic. The *Ratio-Greedy Heuristic* (RGreedy, for short) is our baseline for comparison with the other, more advanced heuristics. It is presented in Fig. 1. Initially, it generates a route ρ that *(1)* goes from s to t, *(2)* visits exactly one object of each set A_i, and *(3)* is as short as possible. The task of computing this initial route is by itself a hard problem, and thus, it is solved heuristically. In our implementation, we used the Infrequent-First Heuristic of [7] for this task.

After the initial step of Line 3, the RGreedy algorithm iteratively increases the probability-of-success of the route, using a greedy approach. While the length of ρ does not exceed the length limit ℓ, the algorithm repeatedly extends ρ by adding

```
Ratio Greedy $((s, t, Q_1, \dots, Q_m), \ell, D, Pr())$
Input: Source location $s$, target location $t$, search queries $Q_1, \dots, Q_m$, limit distance $\ell$, a dataset $D$, a probability function $Pr()$
Output: A route starting at $s$, ending at $t$, not exceeding a length of $\ell$ and attempting to provide the highest probability-of-success
1: for $i = 1$ to $m$ do
2:     $A_i \leftarrow Q_i(D)$
3: $\rho \leftarrow$ ShortestRouteHeuristic$((s, t, Q_1, \dots, Q_m), D)$    // the initial short route
4: $Candidates \leftarrow \{o \mid$ there exists an $i$ such that $length(Add_i(\rho, o)) \leq \ell\}$
5: while $Candidates \neq \emptyset$ do
6:     let $i$ be an index and $o_m$ be an object of $Candidates$ such that for all $o \in$
       $Candidates$ and index $j$, it holds that $Ratio_i(\rho, o_m) \geq Ratio_j(\rho, o)$
7:     $\rho \leftarrow Add_i(\rho, o_m)$
8:     $Candidates \leftarrow \{o \mid$ there exists an $i$ such that $length(Add_i(\rho, o)) \leq \ell\}$
9: return $\rho$
```

Fig. 1. The Ratio-Greedy Heuristic for answering route-search queries under the bounded-length semantics

a new object in each iteration of Line 5. We use $Add_i(\rho, o)$ to denote the route that is obtained by adding the object o between the i-th and the $(i+1)$-st objects of ρ. That is, if $\rho = s, o_1, \dots, o_k, t$ then $Add_i(\rho, o) = s, \dots, o_i, o, o_{i+1}, \dots, t$. When $i = 0$, we add o between s and o_1, and similarly, when $i = k$, we add o between o_k and t. An object o is a *candidate* for extension if after adding o to ρ, the length of the new route does not exceed ℓ, that is, there is an i such that $length(Add_i(\rho, o)) \leq \ell$.

The candidate that is actually added to the current route is the one that gives the largest ratio of the increase in the probability-of-success to the increase in the length of the route. That is, let

$$Ratio_i(\rho, o) = \frac{Pr(Add_i(\rho, o)) - Pr(\rho)}{length(Add_i(\rho, o)) - length(\rho)}$$

be this ratio when inserting o at position i of the current route ρ. Each iteration of Line 5 chooses the maximal $Ratio_i(\rho, o)$ and adds o at position i of ρ. Note that o is added at the position where it causes the smallest increase in the length of the route, because the probability does not depend on the length. Hence, the length limit is not exceeded.

We also experimented with two other criteria for adding objects. One is choosing the object that maximizes the increase in the probability-of-success. The second is picking the object that minimizes the increase in the length. However, in the experiments, these two criteria were found to be inferior to choosing according to the largest ratio.

```
Increase-Decrease $((s, t, Q_1, \dots, Q_m), \ell, D, Pr())$

Input: Source location $s$, target location $t$, search queries $Q_1, \dots, Q_m$, limit distance $\ell$, a dataset $D$, a probability function $Pr()$
Output: A route starting at $s$, ending at $t$, not exceeding a length of $\ell$ and attempting to provide the highest probability-of-success
1: for $i = 1$ to $m$ do
2:     $A_i \leftarrow Q_i(D)$
3: $\rho \leftarrow$ RGreedy$((s, t, Q_1, \dots, Q_m), \ell, D, Pr())$     // $\rho$ is the initial route
4: let $k$ be the number of objects in the current route, i.e., $\rho = s, o_1, \dots, o_k, t$
5: $r \leftarrow \lceil \frac{k}{2} \rceil$
6: while $r > 0$ do
7:     for $i = 0$ to $k - r$ do
8:         let $\rho_i^-$ be the result of removing from $\rho$ the objects $o_{i+1}, \dots, o_{i+r}$, i.e.,
           $$\rho_i^- = \begin{cases} s, o_{r+1}, \dots, o_k, t & : \ i = 0 \\ s, o_1, \dots, o_i, o_{i+r+1}, \dots, t & : \ 0 < i < k - r \\ s, o_1, \dots, o_{k-r}, t & : \ i = k - r \end{cases}$$
9:     let $\rho_m^-$ be a route in $\{\rho_0^-, \dots, \rho_{k-r}^-\}$ whose length is minimal, i.e., for every
       $0 \le i \le k - r$ it holds that $length(\rho_m^-) \le length(\rho_i^-)$
10:    let $(\rho_m^-)^+$ be the result of adding objects to $\rho_m^-$ by applying the increase
       stage of the Ratio-Greedy Heuristic (Lines 4-9 of Fig. 1)
11:    if $Pr((\rho^-)^+) > Pr(\rho)$ then
12:        $\rho \leftarrow (\rho^-)^+$
13:    else
14:        $r \leftarrow r - 1$
15: return $\rho$
```

Fig. 2. The Increase-Decrease Heuristic for answering route-search queries under the bounded-length semantics

Increase-Decrease Heuristic. The Ratio-Greedy Heuristics only adds objects. An object o that is inserted into the route, either in the initial step or during the iterative stage, remains in the route even if its utility is eventually diminished (this may happen when subsequent iterations add many objects that are far away from o). The *Increase-Decrease Heuristic* (Inc-Dec, for short) changes that by considering the option of removing objects from the current route and replacing them with some other objects.

Increase-Decrease is presented in Fig. 2. Initially, Line 3 generates a route using the Ratio-Greedy Heuristic. Then, each iteration of Line 6 picks r adjacent objects as candidates for removal. In particular, Line 9 chooses the r contiguous objects that yield the largest decrease in the length of the current route. These r objects are actually removed if the Ratio-Greedy Heuristic can add some other objects, such that the new route has a higher probability-of-success and does not exceed the length limit (Lines 10–12). If the objects are indeed removed, r is not changed; otherwise, it is decreased by one. In either case, the next iteration of Line 6 tries again to replace r objects. The algorithm terminates when $r = 0$.

Extra-Length Heuristic. The former heuristics are unlikely to add objects that are not close to the initial route, because of the length limit. Yet, in some cases, it may be beneficial to add objects that are far away from the initial route if those objects have high probabilities and are clustered together in a small area. The approach of the *Extra-Length Heuristic* (Ext-Len, for short), presented in Fig. 3, is to start by applying the Ratio-Greedy Heuristic with a *pseudo* length limit that is larger than the actual one (*e.g.*, multiply the given ℓ by five). This makes it possible to generate intermediate routes that visit far-away objects. Clearly, the route obtained in Line 4 is too long. Hence, objects are repeatedly removed in the loop of Line 5. In each iteration, the removed object is the one that gives the largest ratio of the decrease in the length to the decrease in the probability-of-success. For an object o, this ratio is

$$\mathit{RemRatio}(\rho, o) = \frac{\mathit{length}(\rho) - \mathit{length}(\mathit{Remove}(\rho, o))}{\mathit{Pr}(\rho) - \mathit{Pr}(\mathit{Remove}(\rho, o))}$$

where $\mathit{Remove}(\rho, o)$ is the route obtained by removing o from ρ. Since this may lead to removing "too much," a final stage of adding objects using the Ratio-Greedy approach is applied (Line 8).

When using a large length limit, the probability-of-success of the current route can become close to one. Consequently, all the candidates for addition to

```
Extra Length $((s, t, Q_1, \ldots, Q_m), \ell, D, Pr())$
Input: Source location $s$, target location $t$, search queries $Q_1, \ldots, Q_m$, limit distance $\ell$, a dataset $D$, a probability function $Pr()$
Output: A route starting at $s$, ending at $t$, not exceeding a length of $\ell$ and attempting to provide the highest probability-of-success
1: for $i = 1$ to $m$ do
2:     $A_i \leftarrow Q_i(D)$
3: $\ell_{\text{extra}} \leftarrow 5\ell$
4: $\rho \leftarrow$ RGreedy$^{\varepsilon}((s, t, Q_1, \ldots, Q_m), \ell_{\text{extra}}, D, Pr())$    /* $\rho$ is the initial route
   generated by using the Ratio-Greedy Heuristic while guaranteeing that the
   probability-of-success does not exceed $1 - \varepsilon$ */
5: while $length(\rho) > \ell$ do
6:     let $o_r$ be an object in $\rho$ such that $RemRatio(\rho, o_r) \geq RemRatio(\rho, o)$ for
       every object $o$ in $\rho$
7:     $\rho \leftarrow Remove(\rho, o_r)$
8: add objects to $\rho$ by applying the increase stage of the Ratio Greedy Heuristic
   (Lines 4-9 of Fig. 1)
9: return $\rho$
```

Fig. 3. The Extra-Length Heuristic for answering route-search queries under the bounded-length semantics

```
Universal Start $((s, t, Q_1, \ldots, Q_m), \ell, D, Pr())$
Input: Source location $s$, target location $t$, search queries $Q_1, \ldots, Q_m$, limit distance $\ell$, a dataset $D$, a probability function $Pr()$
Output: A route starting at $s$, ending at $t$, not exceeding a length of $\ell$ and attempting to provide the highest probability-of-success
1: for $i = 1$ to $m$ do
2:     $A_i \leftarrow Q_i(D)$
3: $\rho_m \leftarrow s, t$
4: for each $o$ in $D$ do
5:     generate an initial route $\rho = s, d_1, \ldots, d_m, o, t$ where $o$ is an object of $D$
       and $d_1, \ldots, d_m$ are dummy objects all located at $s$, satisfying $d_i \in A_i$, for
       $1 \leq i \leq m$ and with probability-of-success that is a small $\varepsilon$ (i.e., the dummy
       objects do not affect the length of the route and just guarantee that the
       probability-of-success of $\rho$ is not zero, even before adding an object of each
       group among $A_1, \ldots, A_m$)
6:     add objects to $\rho$ by applying the increase stage of the Ratio Greedy Heuristic
       (Lines 4-9 of Fig. 1)
7:     if $Pr(\rho) > Pr(\rho_m)$ then
8:         $\rho_m \leftarrow \rho$
9: return $\rho_m$
```

Fig. 4. The Universal-Start Heuristic for answering route-search queries under the bounded-length semantics

the current route yield similar ratios, and the Ratio-Greedy Heuristic may add objects that have very low probabilities. In order to avoid that, we use a variation RGreedy$^{\varepsilon}$ of the Ratio-Greedy Heuristic that prevents the current route from reaching a probability-of-success that is too close to one. This variant stops when either the given length limit is reached or the probability-of-success exceeds $1-\varepsilon$.

Universal-Start Heuristic. All of the former heuristics may fail to consider some of the objects that are needed for generating a route that is (close to) the best. The *Universal-Start Heuristic* (UStart, for short), shown in Fig. 4, surmounts this problem by applying the Ratio-Greedy Heuristic multiple times. In particular, in the loop of Line 4, UStart considers all the objects o that are within the length limit. For a given o, Line 5 creates an initial route consisting of the start location, the target location and o itself. Line 6 applies the Ratio-Greedy Heuristic to this initial route. The output is the route with the highest probability-of-success. In order for the initial route to have a nonzero probability-of-success, Line 5 adds dummy objects that do not change the length and have an ε effect on the probability, where ε is a very small constant. These placeholders are removed after adding the relevant objects, *i.e.*, the dummy object of A_i is deleted from the current route when an object of A_i is added to the route.

4.2 Heuristics for the Bounded-Probability Semantics

With some minor modifications, the above heuristics for the bounded-length semantics can also be used for the bounded-probability semantics. We now describe these modifications.

In the case of the the Ratio-Greedy Heuristic, we only need to change the termination condition. In other words, this heuristic repeatedly extends the current route, as described earlier, until the probability-of-success of the current route exceeds the threshold p. Clearly, this modification applies to the original Ratio-Greedy Heuristic (Fig. 1) as well as to the application of this heuristic by the other three algorithms.

The Extra-Length Heuristic is changed into an Extra-Probability Heuristic as follows. We first generate an initial route that has a much higher probability than the given threshold p. We do it by multiplying p by some constant and then applying the modified Ratio-Greedy Heuristic. In addition, we change the termination condition in the phase of removing objects. This phase continues while the probability-of-success is greater than the probability bound p.

No changes are needed in the Increase-Decrease and the Universal-Start Heuristics, except for the obvious fact that they apply the Ratio-Greedy Heuristics with the modification described above.

4.3 Complexity Analysis

We now discuss the complexity of the four heuristics. For simplicity, we consider the heuristics only under the bounded-length semantics.

First, we note that some of the objects of the underlying dataset may not be relevant to the computation of a given route-search query. Specifically, given a query $R = (s, t, \mathcal{Q})$ and a length limit ℓ, only the objects in the elliptic area $\{u \mid \mathit{distance}(s, u) + \mathit{distance}(u, t) \leq \ell\}$ are relevant to the computations. Objects outside this area cannot be in the result. When a spatial data structure exists for the dataset over which R is computed, it can be used for retrieving the relevant objects before the computation. Otherwise, the algorithms can start by reading the entire dataset and filtering out irrelevant objects. Hence, in the rest of this section, we will consider D to be a dataset of merely n relevant objects.

In the Ratio-Greedy Heuristic, the complexity of constructing the initial route depends on the algorithm being used. The iterative stage has $O(n^3)$ time complexity. To see that, first note that there are at most n iterations of adding an object to the route. In each iteration, at most n objects are considered as candidates for addition. The computation where to add an object and the computation of the probability are proportional to the length of the sequence, which is at most n. Note that if we consider k to be the maximal number of objects in a route, then the time complexity is actually $O(nk^2)$.

For the Increase-Decrease Heuristic, again our analysis considers only the phase after the initial route has been computed. In this algorithm, the main iteration is performed at most 2^n times. To see why, observe that in each iteration, we replace a subset of the route by a new subset. Since the probability of the route constantly increases, no two routes can have exactly the same set of objects. Hence, there are at most 2^n possible replacements. Now, in each iteration, the iterative stage of the Ratio-Greedy Heuristic is executed (and as shown above, it has $O(nk^2)$ time complexity). Thus, the time complexity of this heuristic is $O(n2^nk^2)$.

The time complexity of the Extra-Length Heuristic is similar to that of the Ratio-Greedy Heuristic, *i.e.*, $O(nk^2)$. However, the former does more iterations than the latter, because it uses a larger length limit and, moreover, it has an additional phase of removing objects. Finally, the Universal-Start Heuristic applies the Ratio-Greedy Heuristic n times and, thus, has $O(n^2k^2)$ time complexity.

5 Experiments

We have tested our algorithms over both synthetically-generated datasets and real-world datasets. The goal of our experiments was to compare our methods in terms of efficiency and the quality of the results. The experiments were conducted on a PC equipped with a Core 2 Duo processor 2.13 GHz (E6400), 2 GB of main memory and Windows XP Professional operating system.

5.1 Tests on Real-World Data

We extracted real-world data from a digital map of the City of Tel-Aviv. A fragment of that data is shown in Fig. 5. Specifically, we used objects of the "Point Of Interest" (POI) layer of the map. This layer represents many different types of geographical entities. The extracted dataset comprises 103 objects of three different types (20 cinemas, 29 hotels, 54 post offices). As a query, we used a particular choice of source and target locations, and assumed that all the objects are in the answer to the query. The objects received probabilities that are normally distributed, with mean of 0.5. We ran additional experiments on more queries as well as another dataset with five types of objects, and the results were similar.

Evaluation under the Bounded-Length Semantics. Fig. 7 and Fig. 8 present the test results for the four heuristics over the real-world dataset, under the bounded-length semantics. Fig. 7 shows the quality of the answer of each algorithm as a function of the length limit ℓ. In this graph, the x-axis shows the factor by which the length limit ℓ is larger than the length of the initial route created by RGeedy. Recall that this initial route has exactly one object from each A_i and is constructed by the Infrequent-First Heuristic of [7]. For instance, when x is 1.8, it means that $\ell = 1.8 \times a$, where a is the length of the initial route. The y-axis is the probability-of-success of the answer. It can be seen that UStart provides the best answers. Ext-Len provides answers that are almost as good. The quality of the answers of RGreedy are the worst among the four heuristics.

Fig. 8 presents the running times of the methods as a function of the length limit ℓ. The x-axis is similar to the x-axis in Fig. 7. The y-axis shows the running time of the computation, in milliseconds. RGreedy is the most efficient. Ext-Len

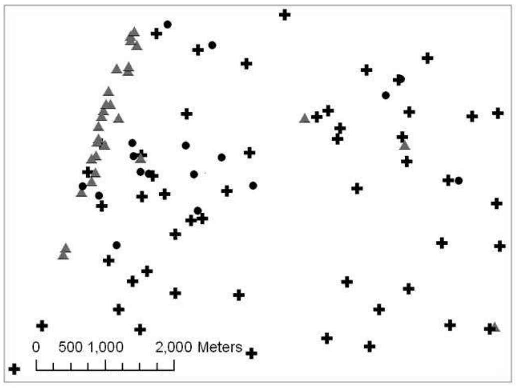

Fig. 5. A fragment of the real-world dataset

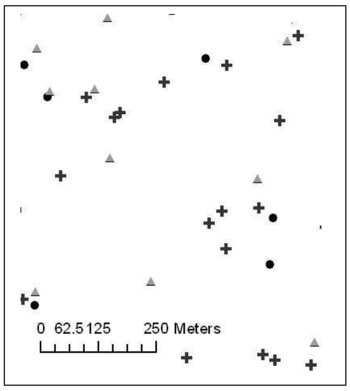

Fig. 6. A fragment of the synthetic dataset

is almost as efficient as RGreedy. Note that its running time does not change as ℓ grows, because even when ℓ is small, the computation is with respect to a larger length (i.e., 5ℓ). The Inc-Dec heuristic is efficient for small values of ℓ, but its running time climbs sharply as ℓ grows, and then it drops again. The reason for that is that the size of the replaced subsets, as well as their number, grows when the route becomes longer. However, the probability of the generated route also grows when the length limit increases. In the implementation, Inc-Dec stops when it finds a route with a probability that is very close to one. This happens early when the length limit is large and, therefore, the running time drops.

The running times of UStart are not shown in Fig. 8, because they are much greater than those of the other methods. In order to compare them to the running times of the other three methods, they are presented in Table 1. For large values of the length limit, the running time of UStart drops, because UStart quickly finds a route with a probability that is very close to one (and in the implementation, UStart stops when that happens).

Table 1. The running times (milliseconds) under the bounded-length semantics

Length	RGreedy	Inc-Dec	Ext-Len	UStart
1.01	2	7	81	34
1.4	7	43	90	551
1.8	10	61	82	2193
2.2	24	331	100	209
2.6	45	179	80	178

Evaluation under the Bounded-Probability Semantics. Fig. 9 and Fig. 10 present the test results of the algorithms over the real-world dataset, under the bounded-probability semantics. Fig. 9 shows the length of the answer as a function of the probability threshold. Not surprisingly, these results are analogous

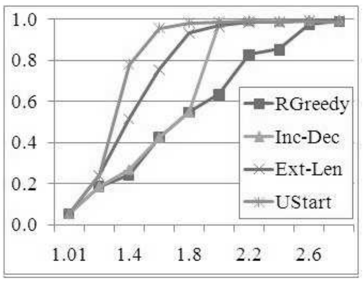

Fig. 7. Probability versus length limit

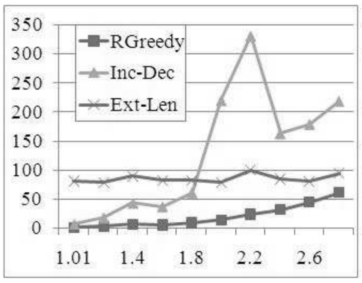

Fig. 8. Running times (millisecond) versus length limit

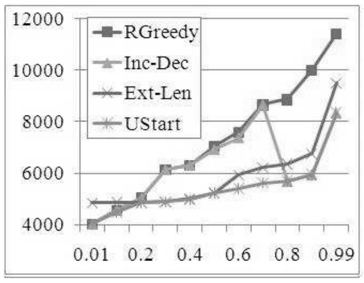

Fig. 9. Length versus probability bound

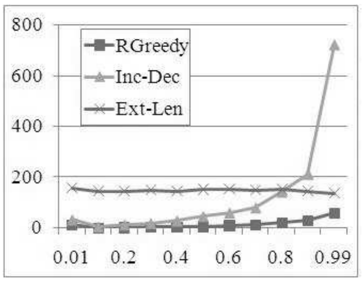

Fig. 10. Running times (millisecond) versus probability bound

to the results under the bounded-length semantics. UStart provides the best answers, Ext-Len is almost as good as UStart, while RGreedy provides the longest route among the four methods. The test results of Inc-Dec are similar to those of RGreedy for low probabilities, and similar to those of UStart for high probabilities. The reason for that is that when the probability threshold increases, the generated route becomes longer. Hence, the size and number of the replaced subsets is greater, thereby creating more opportunities for improving the route.

Fig. 10 shows the running time of each methods as a function of the probability threshold. Here, as well, the results are similar to the results under the bounded-length semantics. RGreedy is the most efficient. Ext-Len is almost as efficient as RGreedy and its running time does not change much as a function of the probability threshold. Inc-Dec is very efficient for small and medium probability thresholds, but is very inefficient in the case of high thresholds, because it makes many iterations of replacement in this case.

5.2 Tests on Synthetic Data

In order to have control over our experiments, and for testing our methods over datasets with specific properties, we used synthetically generated datasets. In a synthetic dataset, we have control over the distribution of the locations of objects

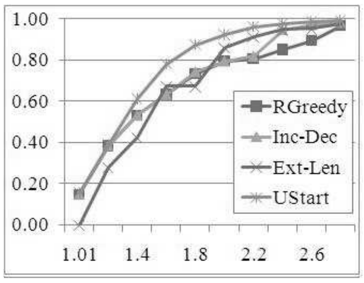

Fig. 11. Probability versus length bound

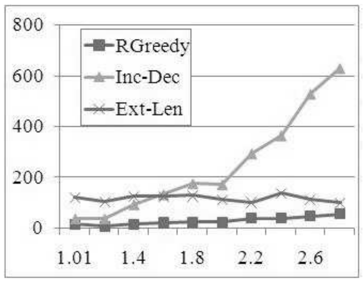

Fig. 12. Running times (milliseconds) versus length bound

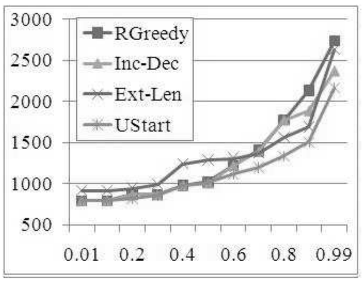

Fig. 13. Length versus probability bound

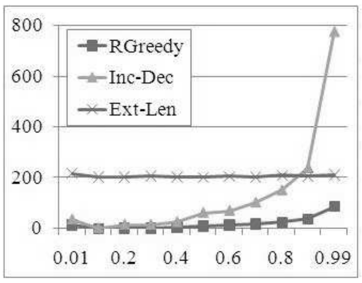

Fig. 14. Running times (milliseconds) versus probability bound

in the area of the map, the way that the objects are partitioned into sets, etc. For generating synthetic datasets, we implemented a random-dataset generator. Our generator is a two-step process. First, the objects are generated. The locations of the objects are randomly chosen according to a given distribution, in a square area. In the second step, we partition the objects into sets and a probability value is attached to each object. The partitioning of objects into sets can be uniform or according to a distribution specified by the user.

The user provides the following parameters to the dataset generator. The number of objects, the size of the square area in which the objects are located and the minimal distance between objects; for simulating search results, the user provides the distribution of object probabilities, and the distribution of the size of the sets in the partition. These parameters allow a user to generate tests with different sizes of datasets and different partitions of the datasets into sets.

We conducted experiments over several synthetically generated datasets and multiple choices of queries. We present a few, typical results. Figures 11, 12, 13 and 14 are for a particular query and a dataset of 150 objects distributed over an area of 1300 $\times$ 1300 square meters (a fragment of this dataset is shown in Fig. 6). The objects in this dataset are partitioned into three types having 25, 50 and 75 objects. All the objects are in the answer to the query.

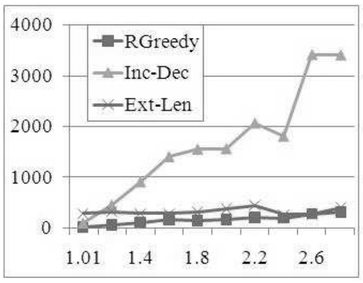

Fig. 15. Running times (millisecond) versus length bound over large dataset

Length	RGreedy	Inc-Dec	Ext-Len	UStart
1.01	14	75	191	216
1.4	118	898	227	11333
1.8	159	1546	407	47521
2.2	216	2045	602	8965
2.6	282	3392	579	1858

Fig. 16. The time for computing a route over dataset with 389 features

Fig. 11 shows the probability-of-success of the answer as a function of the length limit, under the bounded-length semantics. Fig. 12 depicts the running time as a function of the length limit, under the bounded-length semantics. The differences between the methods are not as significant as in the case of real-world data. The reason for that is the uniform distribution of the objects in the area that is queried. Also note that the Ext-Len method usually provides answers that are almost the best, and it does so in time that is almost the shortest. However, it may fail, as can be seen in Fig. 11 when the length limit is very small. In this case, Ext-Len returns zero as the probability of success, because during the removal stage, it removes all the objects of one of the sets.

The results over the synthetic data under the bounded-probability semantics are depicted in Fig. 13 and in Fig. 14. In general, the results over the synthetic data show that Inc-Dec efficiently computes good answers when the bound (of either the length or the probability) is small. For medium or large bounds, Ext-Len is efficient and provides good answers, in comparison to the other methods.

We also tested the scalability of the four heuristics. For that, we used a larger dataset with 389 objects and groups of sizes 31, 169 and 189. The results, which are summarized in Fig. 15 and Fig. 16, show that RGreedy and Ext-Len are scalable. Inc-Dec remains efficient only when the length limit is not high.

6 Conclusion

In this paper, we investigated the problem of route search over probabilistic datasets. We explained why algorithms and heuristics for non-probabilistic route search are inapplicable to probabilistic data, and then, we gave four heuristics for the problem. We showed how these heuristics can be applied under the two proposed semantics. In these heuristics, there is a tradeoff between the efficiency and the quality of the answers. Our experiments over both real-world and synthetically-generated data illustrated this tradeoff and showed that the UStart heuristic provides the best answers at the cost of a relatively long computation time. RGreedy is the most efficient method; however, usually its answers are inferior to those of the other methods. Inc-Dec provides a good tradeoff between the computation time and the quality of the answers when the bounds (i.e., the length limit and the probability threshold) are small. Finally, Ext-Len provides a good tradeoff between the computation time and the quality of the answers in all cases, except when the bounds are small. For future work, we plan to investigate how to use these algorithms in an adaptive way in cases where the user can send or receive information while traveling.

References

1. Chao, I., Golden, B., Wasil, E.: A fast and effective heuristic for the orienteering problem. European Journal of Operational Research 88(3), 475–489 (1996)
2. Chao, I., Golden, B., Wasil, E.: The team orienteering problem. European Journal of Operational Research 88, 464–474 (1996)
3. Chen, H., Ku, W.S., Sun, M.T., Zimmermann, R.: The multi-rule partial sequenced route query. In: GIS, pp. 1–10 (2008)
4. Golden, B., Wang, Q., Liu, L.: A multifaceted heuristic for the orienteering problem. Naval Research Logistics 35, 359–366 (1988)
5. Golden, B.L., Levy, L., Vohra, R.: The orienteering problem. Naval Research Logistics 34, 307–318 (1987)
6. Huang, X., Jensen, C.S.: In-route skyline querying for location-based services. In: Kwon, Y.-J., Bouju, A., Claramunt, C. (eds.) W2GIS 2004. LNCS, vol. 3428, pp. 120–135. Springer, Heidelberg (2005)
7. Kanza, Y., Safra, E., Sagiv, Y., Doytsher, Y.: Heuristic algorithms for route-search queries over geographical data. In: GIS, pp. 1–10 (2008)
8. Keller, P.C.: Algorithms to solve the orienteering problem: A comparison. European Journal of Operational Research 41, 224–231 (1989)
9. Leifer, A.C., Rosenwein, M.S.: Strong linear programming relaxations for the orienteering problem. European J. of Operational Research 73, 517–523 (1994)
10. Li, F., Cheng, D., Hadjieleftheriou, M., Kollios, G., Teng, S.H.: On trip planning queries in spatial databases. In: Bauzer Medeiros, C., Egenhofer, M.J., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 273–290. Springer, Heidelberg (2005)
11. Ma, X., Shekhar, S., Xiong, H., Zhang, P.: Exploiting a page-level upper bound for multi-type nearest neighbor queries. In: GIS, pp. 179–186 (2006)
12. Ramesh, R., Yoon, Y., Karwan, M.: An optimal algorithm for the orienteering tour problem. ORSA Journal on Computing 4(2), 155–165 (1992)
13. Safra, E., Kanza, Y., Dolev, N., Sagiv, Y., Doytsher, Y.: Computing a k-route over uncertain geographical data. In: Papadias, D., Zhang, D., Kollios, G. (eds.) SSTD 2007. LNCS, vol. 4605, pp. 276–293. Springer, Heidelberg (2007)
14. Sharifzadeh, M., Kolahdouzan, M.R., Shahabi, C.: Optimal sequenced route query. The VLDB Journal 17(8), 765–787 (2008)
15. Terrovitis, M., Bakiras, S., Papadias, D., Mouratidis, K.: Constrained shortest path computation. In: Bauzer Medeiros, C., Egenhofer, M.J., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 181–199. Springer, Heidelberg (2005)
16. Tsiligirides, T.: Heuristic methods applied to orienteering. Journal of the Operational Research Society 35(9), 797–809 (1984)