An Interactive Approach to Route Search

Yaron Kanza[*]
Technion
kanza@cs.technion.ac.il

Roy Levin[*]
Technion
royl@cs.technion.ac.il

Eliyahu Safra
safraeli@gmail.com

Yehoshua Sagiv[†]
Hebrew University
sagiv@cs.huji.ac.il

ABSTRACT

In a *probabilistic route search*, there is a start location, a target location, and search queries $Q_1, \ldots, Q_n$. Each Q_i has an answer set A_i consisting of geo-spatial objects and their probabilities. The probability of an object $o \in A_i$ specifies the likelihood that o satisfies Q_i. The goal is to compute a route that is short and yet has a high probability of satisfying all the Q_i. This paper investigates interactive route search. Upon arrival at each object, the user provides feedback specifying whether the object satisfies its corresponding query. The goal is to compute the next object to be visited, based on the feedback. Several heuristic algorithms are given and compared experimentally.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*

General Terms

Algorithms, Experimentation

Keywords

Geographic information system, route, path, search, queries, probabilistic data, heuristic algorithms

1. INTRODUCTION

The goal of a route search is to find a path that passes through several objects of different types that are relevant to the user. For each type, the user specifies a search query that identifies objects of that type that may satisfy her needs (e.g., a seafood restaurant with moderate prices). In addition, there is a start location s (e.g., the user's office) and a target location t (e.g., her home). The goal is to find the shortest route from s to t that visits one object of each type.

Usually, the objects returned by a search query are not certain answers. That is, there is some probability (of less than 100%) that an object actually satisfies the user's needs. For example, when the user arrives at a gift shop, she may not find a gift that she likes. This uncertainty is modeled by assigning a probability to each object that is returned as an answer to a search query.

Since objects have probabilities, an effective route may have to include more than one object of each type, in order to guarantee a sufficiently high level of success. Thus, the desire to have the shortest possible route conflicts with the necessity to guarantee a high probability of satisfying the user's needs. Semantics and algorithms for route search over probabilistic data were investigated in earlier work [3].

In this paper, we deal with *interactive route search*. That is, upon arriving at each object on her route, the user informs the system whether the object satisfies her needs. Based on that feedback, the system computes the next object to be visited by the user. Interactive route search fits the ubiquitousness of hand-held devices in our technologically driven era. Moreover, computing a route interactively is likely to reduce the distance covered by the user.

In earlier work on route search, they either dealt with non-probabilistic datasets (e.g., [1, 2, 5, 7, 8]) or gave non-interactive algorithms (e.g., [6, 4]). Consequently, the results of earlier work cannot be used to solve the interactive route-search problem that we address in this paper. Our contribution is in giving the first algorithms for this problem and showing their effectiveness.

In some cases, the user may want to impose order constraints on the types of objects to be visited. For example, she wants to draw cash from an ATM before going to a restaurant. We will deal with order constraints in a forthcoming paper.

2. ROUTE-SEARCH QUERIES

A *geo-spatial dataset* is a collection O of objects. Each object represents a real-world geographical entity and its location is the same as that of the entity. 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, that is, different

[*]The work of these authors 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).

objects have different locations. Distances between objects are not necessarily Euclidean; rather, they can be computed from a given road network. We use $dist(o, o')$ to denote the distance from o to o'.

A *search query* specifies a collection of objects that are of interest to the user. We do not consider a specific syntax or semantics of search queries. For example, a search query Q_i could be stated as a list of keywords (e.g., `Vegetarian Japanese Restaurant`) and some constraints (e.g., `rank` $\geq$ 3). Our only assumption is that the evaluation of Q_i returns an *answer set* A_i comprising objects and their probabilities. The probability of an object $o \in A_i$ specifies the likelihood that o *satisfies* Q_i. Discussing how the probabilities are computed is beyond the scope of the paper.

Route-search queries are generated by combining several search queries that specify different types of objects that the user would like to visit. We represent a route-search query as a triplet $R = (s, t, \mathcal{Q})$, where s is a *start location*, t is a *target location*, and $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ is a set of search queries. We assume that the answer set $A_1, \ldots, A_m$ of $Q_1, \ldots, Q_m$, respectively, are pairwise disjoint (this entails no loss of generality).

3. INTERACTIVE ALGORITHMS

We now describe interactive algorithms for solving a route-search query $R = (s, t, \mathcal{Q})$, where $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ and $A_1, \ldots, A_m$ are the answer sets of $Q_1, \ldots, Q_m$, respectively. All the algorithms operate over the objects of $A_1, \ldots, A_m$, and they compute a route by iteratively increasing a partial sequence σ. Initially, the partial sequence comprises only the start location, namely, $\sigma = s$. On each iteration, the algorithms computes the next object o_k to be visited; thus, o_k is added at the end of σ. When arriving at o_k, the user provides a feedback, denoted by $o\text{-}sat(o_k)$. If o_k actually satisfies the corresponding search query (i.e., the Q_i, such that $o_k \in A_i$), then $o\text{-}sat(o_k)$ is `true`; otherwise $o\text{-}sat(o_k)$ is `false`.

Let $\sigma = s, o_1, \ldots, o_k$ be the partial sequence visited thus far. The set of *unsatisfied queries* of R with respect to (abbr. w.r.t.) σ, denoted by $q\text{-}unsat_R(\sigma)$, comprises all the Q_i, such that σ has no object that satisfies Q_i; that is,

$$q\text{-}unsat_R(\sigma) = \{Q_i \mid Q_i \in \mathcal{Q} \text{ and } \neg\exists o(o \in \sigma \wedge o \in A_i \wedge o\text{-}sat(o))\}.$$

Each of our algorithms chooses the next object to be visited from the set of *candidate objects* w.r.t. σ, denoted by $candidates_R(\sigma)$. This set comprises the objects that neither have been visited nor belong to answer sets of search queries that have already been satisfied; that is,

$$candidates_R(\sigma) = \{o \mid \exists Q_i(Q_i \in q\text{-}unsat_R(\sigma) \wedge \wedge\, o \in A_i \wedge o \notin \sigma)\}.$$

Observe that if $A_j \cap candidates_R(\sigma) = \emptyset$ for some $Q_j \in q\text{-}unsat_R(\sigma)$, then σ cannot be completed to a route that satisfies all the search queries. In addition, if $q\text{-}unsat_R(\sigma) = \emptyset$, then all the queries of $\mathcal{Q}$ have been satisfied, and hence, t is the next destination and the computation ends.

3.1 Naive and Oriented Greedy Heuristics

The *naive greedy heuristic* chooses the candidate object that is closest to the current location l (where l is either s or some o_k). The naive greedy heuristic is simple and efficient. However, it suffers from the drawback of ignoring the location of the target t. Consequently, it may compute a route that drifts far away from t and is unnecessarily long, due to the distance from the last object to t. A possible solution is to choose the next object o' based on the combined distance of o' from both the current location and t. This approach is likely to compute a route in the general direction toward t. But the route might progress too fast toward t, that is, within a few steps, the route will reach objects near t, even when there are many relevant objects in the vicinity of s and only a few near t.

The *oriented greedy heuristic* assuages the above problems by choosing the next object o' so that it will be near the current location as well as close to the straight line from s to t. In order to do so, this heuristic chooses the candidate object o' that minimizes the sum of distances $dist(l, o') + dist(s, o') + dist(o', t)$, where l is the current location. In the sequel, the oriented greedy heuristic is called the *Greedy* algorithm.

3.2 Optimistic Approach

The greedy approach has a local scope. Namely, it picks the next object o' without taking into account the likelihood that this choice would eventually lead to the shortest route that satisfies all the remaining queries. The *optimistic approach* takes a global view by looking for the shortest route (from the current location to t) that might satisfy all the remaining search queries. This approach is optimistic in the sense that it ignores the probabilities and assumes that objects of the answer sets definitely satisfy their corresponding queries.

The first step is to compute the shortest route from s to t that passes through one object of each answer set. Unfortunately, this is an NP-hard problem [4]. Therefore, we use the infrequent-first heuristic (IFH) of [4] for this task. The user travels along the route computed by IFH until she encounters the first object o_k that does not satisfy its corresponding search query. When that happens, the optimistic algorithm uses IFH to compute a new route (from the current location o_k to t) that satisfies all the remaining search queries. Observe that the new route passes through exactly one object of each set $A_{j_i} \cap candidates_R(\sigma)$, where $1 \leq i \leq d$ and $A_{j_1}, \ldots, A_{j_d}$ are the answer sets corresponding to the search queries of $q\text{-}unsat_R(\sigma)$.

3.3 Minimizing the Expected Distance (MED)

The optimistic approach employs a best-case scenario by assuming that objects definitely satisfy their corresponding search queries. A more realistic approach is to use an average-case analysis. The main idea is to choose the next object based on the expected, rather than the shortest, distance that still remains to be traveled. To formalize this notion, let s be the current location and consider an object o. The following is a recursive definition of the expected distance to be covered, given that o is the next object to be visited. We use ℓ_s and ℓ_f to denote the expected distances from o to the target location, depending on whether o succeeds (i.e., satisfies its corresponding query) or fails, respectively. Thus, given that o is the next object, the expected distance from the current to the target location is the following sum.

$$dist(s, o) + prob(o) \cdot \ell_s + (1 - prob(o)) \cdot \ell_f \tag{1}$$

In the MED approach, the next object o to be visited is one that minimizes the above sum.

Computing the expected distance for an object o is not easy. It may involve an exponential number of possible routes, and for each one, we need to keep the history in order to avoid visiting the same object more than once. Hence, we use heuristics that estimate the expected distance, rather than compute it precisely.

For the sake of efficiency, the MED algorithm considers only k candidate objects, namely, the top-k according to the oriented greedy heuristic (w.r.t. the current location). The parameter k is provided by the user and in our experiments was usually equal to four. The MED algorithm estimates the distances ℓ_s and ℓ_f using IFH. Thus, the overall estimation obtained by Equation (1) is crude, because it only takes into account the probability of o, but not those of the objects visited after o. Nonetheless, we do it in this way for efficiency considerations.

3.4 Letting The Probability Affect the Route

When computing a route, most of the above algorithms consider only the distances between objects, but ignore the probabilities. One way to add the effect of the probabilities is by changing the distance function as follows. For every two objects o_1 and o_2, the distance function $dist_p(o_1, o_2)$ is defined to be $dist(o_1, o_2)/prob(o_2)$. We can now use $dist_p$ instead of $dist$. This increases the distance to objects with a low probability of success in a manner that is inversely proportional to the probability.

4. EXPERIMENTS

In order to examine the effectiveness and efficiency of our methods, we tested them over both syntactic and real-world data in a variety of cases. Due to lack of space, however, we only describe the experiments over the real-world data and discuss the main conclusions.

The real-world data that we used in our experiments is part of a digital map, of the city Tel-Aviv, that has been generated by the Mapa company. In our tests, we used the "Point Of Interest" (POI) layer of the map. The objects in this layer represent many different types of geographical entities. We extracted from the map 628 objects of seven different types (20 cinemas, 29 hotels, 31 pedestrian bridges, 54 post offices, 136 pharmacies, 169 parking lots and 189 synagogues). In the experiments, we tested route-search queries R where the number of search queries in $\mathcal{Q}$ is between three to seven.

In order to simulate interactive scenarios, the satisfaction of each visited object was chosen randomly, when the object was visited, according to the probability of the object. Since we wanted to prevent extreme cases, we ran every query 100 times, where in each run, different random choices were made for the objects, and the results were averaged.

The Greedy algorithm (i.e., oriented greedy) and the Optimistic algorithm have two versions—one that uses the actual distances between objects and another that uses weighted distances (in the weighted version, for each edge of the network, the actual distance is divided by the probability of the target object). We denote the actual and weighted versions of Greedy by *aGre* and *wGre*, in correspondence, and by *aOpt* and *wOpt* for Optimistic. In our experiments, *wOpt* always provides better results than *aOpt*, and *wGre* almost always provides better results than *aGre*. The reason for

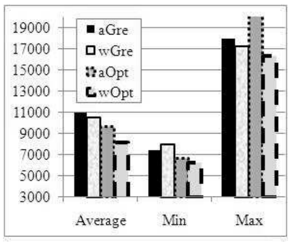

Figure 1: Normally distributed probabilities

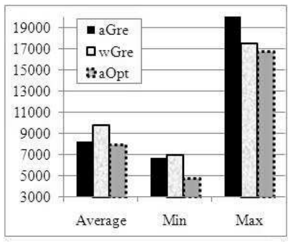

Figure 2: Five answer sets with high probabilities and two with low probabilities

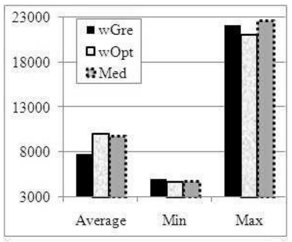

Figure 3: All probabilities are low

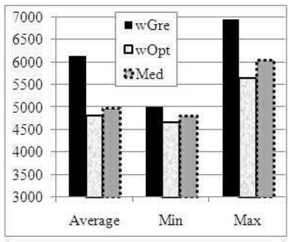

Figure 4: All probabilities are high

this is that the weighted versions prevent the user from going to many objects whose probability is low. Instead, the user goes directly to objects with high probability, *i.e.,* to objects whose chance of satisfying the query is high. The results are presented in Figure 1.

The graph in Figure 1 was produced by combining the results of three different route-search queries, all three comprise seven search queries, but each one with a different pair of start and end points. For each query, the algorithms *aGre*, *wGre*, *aOpt* and *wOpt* were run 100 times over the Tel-Aviv dataset. The probabilities of the objects in the dataset were normally distributed with mean of 69.7 percent and a standard deviation of 9.98 percent.

Figure 1 shows the average length of the produced routes, for the 300 runs (100 runs per query). It also shows the length of the shortest route (the columns with the caption Min) and the length of the longest route (the columns with the caption Max) in these 300 runs. The graph shows that the average length of the computed routes is smaller for *wGre* than for *aGre*, and the same holds for *wOpt* and *aOpt*. Note that this does not guarantee that for a single run of a single query, the route produced by *wGre* will be shorter than the route produced by *aGre*, and similarly for *wOpt* and *aOpt*. The Min and Max columns provide an indication to this.

One case where *aGre* is better than *wGre* is when the objects of large answer sets have high probabilities while objects of small answer sets have low probabilities. This is because being near an object of a small answer set happens less frequently than being in proximity to an objects of a large answer set. However, not giving precedence to visiting infrequent objects may lead to the need to visit such objects when none of them is in proximity to the current location. The results of the algorithms in such case are shown by the graph in Figure 2. This graph shows the results of queries

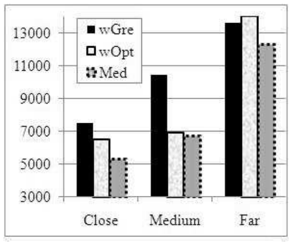

Figure 5: Different start and end locations, four answer sets have high probabilities and three have low probabilities

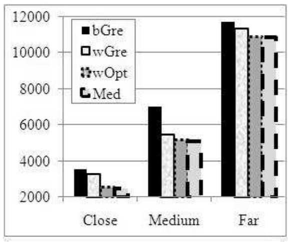

Figure 6: Different start and end locations, normally distributed probabilities

over a dataset where for five large answer sets the probability of objects is high (0.9) and for two small answer sets, the probability of the objects is low (0.3). It can be seen that in such case *aGre* is better than *wGre*.

When all the probabilities are low, the Greedy algorithm is, on the average, better than the other algorithms. The reason for this is that when the probabilities are low, the route should go via many objects since most of the objects fail to satisfy the query. The Optimistic approach and MED try to find a short route to t and only visit several objects on the way. These algorithm reach t and then need to go back to unvisited objects. The Greedy algorithm, in comparison, goes from an object to the next nearest object and, thus, visits many objects on the way to t. This reduces the number of cases where the algorithm needs to go back to unvisited objects. Figure 3 shows the results of an experiment where the probabilities are low. In this experiment, all the objects of all the seven datasets have a probability of 0.3. It can be seen that the average length of a route produced by *wGre* is smaller than the average length of the routes computed by *wOpt* or *MED*. (Note that in this experiment, all the objects have the same probability and, thus, *aGre* is the same as *wGre*, and *aOpt* is identical to *wOpt*.)

When all the probabilities are high, the Optimistic algorithm outperforms the Greedy and MED algorithms. This is because for high probabilities the objects satisfy the query in most of the cases, and thus, most of the time the shortest pre-answer succeeds to satisfy all the queries. The Optimistic algorithm, which follows the shortest pre-answer, computes in all these cases routes that are at least as short as the routes computed by Greedy or MED. An extreme case is when all the objects have a probability that is equal to 1. In this case, the shortest pre-answer, which the Optimistic algorithm computes, is the optimal solution, *i.e.,* a route that is shorter than any other possible route. Figure 4 shows the results of an experiment over a dataset in which all the objects have a high probability of 0.9. Indeed, in this case the average length of the routes computed by *wOpt* is smaller than the average length of the routes computed by *wGre* or *MED*.

In almost all cases, MED computes results that are at least as good as the other algorithms. We have seen that in the results of many experiments and we present one of these experiments in Figure 5. In this experiment, the route-search query was executed over a dataset in which the objects of four answer sets received very high probabilities and the objects of three answer sets received low probabilities. The test was conducted for queries where s and t are near each other (the columns with the caption Close), far from each other (the columns with caption Far), and neither too close to nor too far away from each other (the columns whose caption is Medium). In all of these cases, *MED* is better than *wOpt* and is much better than *wGre*.

We have also compared our algorithms to Naive Greedy. The results are presented in Figure 6, where Naive Greedy is called *bGre*. In this experiment, the algorithms were executed over a dataset with three answer sets, where the probabilities are normally distributed with mean of 69.7 percent and a standard deviation of 9.98 percent. The experiment tested queries with various distances between the start location s and the target location t. The experiment shows that in all the cases, *wGre*, *wOpt* and *MED* are much better that the Naive Greedy.

5. CONCLUSION

We presented three algorithms for interactive route search. The Greedy algorithm provides the best results for the case where all the probabilities of the objects are low. The Optimistic algorithm provides the best results when all the probabilities are high. In all the other cases, including the case where some objects have high probabilities and some objects have low probabilities, the MED algorithm provides the best results. All the algorithms are practical and efficient; in particular, the time needed to compute the next object is a few milliseconds, several tens of milliseconds and a hundred milliseconds for the Greedy, Optimistic and MED algorithms, respectively.

An important generalization of probabilistic route search is when there are order constraints on the types of objects to be visited (e.g., an ATM should be visited before a restaurant). We have already developed algorithms for this generalization and will introduce them in a forthcoming paper.

For future work, we plan to improve the estimation of the expected distance in the MED algorithm, without sacrificing efficiency. We also intend to investigate the problem of computing route-search queries in the presence of time constraints and traffic conditions.

6. REFERENCES

[1] H. Chen, W.-S. Ku, M.-T. Sun, and R. Zimmermann. The multi-rule partial sequenced route query. In *GIS*, pages 1–10, 2008.

[2] X. Huang and C. Jensen. In-route skyline querying for location-based services. In *W2GIS*, pages 120–135, 2004.

[3] Y. Kanza, E. Safra, and Y. Sagiv. Route search over probabilistic geospatial data. In *SSTD*, pages 153–170, 2009.

[4] Y. Kanza, E. Safra, Y. Sagiv, and Y. Doytsher. Heuristic algorithms for route-search queries over geographical data. In *GIS*, pages 1–10, 2008.

[5] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S. Teng. On trip planning queries in spatial databases. In *SSTD*, pages 273–290, 2005.

[6] E. Safra, Y. Kanza, N. Dolev, Y. Sagiv, and Y. Doytsher. Computing a k-route over uncertain geographical data. In *SSTD*, pages 276–293, 2007.

[7] M. Sharifzadeh, M. R. Kolahdouzan, and C. Shahabi. Optimal sequenced route query. *VLDBJ*, 17(8):765–787, 2008.

[8] M. Terrovitis, S. Bakiras, D. Papadias, and K. Mouratidis. Constrained shortest path computation. In *SSTD*, pages 181–199, 2005.