Optimal Route Search with the Coverage of Users' Preferences

Yifeng Zeng[1] **Xuefeng Chen**[2] **Xin Cao**[3] **Shengchao Qin**[1] **Marc Cavazza**[1] **Yanping Xiang**[2]
[1]School of Computing, Teesside University, UK, {y.zeng, s.qin, m.o.cavazza}@tees.ac.uk
[2]School of Computer Science and Engineering,
University of Electronic Science and Technology of China, China,
{cxflovechina, xiangyanping}@gmail.com
[3]School of Electronics, Electrical Engineering and Computer Science,
Queen's University Belfast, UK,
x.cao@qub.ac.uk

Abstract

The preferences of users are important in route search and planning. For example, when a user plans a trip within a city, their preferences can be expressed as keywords *shopping mall*, *restaurant*, and *museum*, with weights 0.5, 0.4, and 0.1, respectively. The resulting route should best satisfy their weighted preferences. In this paper, we take into account the weighted user preferences in route search, and present a keyword coverage problem, which finds an optimal route from a source location to a target location such that the keyword coverage is optimized and that the budget score satisfies a specified constraint. We prove that this problem is NP-hard. To solve this complex problem, we propose an optimal route search based on an A* variant for which we have defined an admissible heuristic function. The experiments conducted on real-world datasets demonstrate both the efficiency and accuracy of our proposed algorithms.

1 Introduction

It is important to consider preferences of a user when a route plan is required for their trip to a city. The user's preferences can be expressed by a set of keywords, e.g., *shopping mall*, *restaurant*, *museum*, and such locations can also be weighted differently. For example, one may enjoy shopping and local restaurants more and it is less painful for them to drop out of a museum visit than a visit to the museum in the trip, in which case *shopping mall*, *restaurant*, *museum* may be weighted, for example, by 0.5, 0.4, and 0.1, respectively. To best satisfy the user's needs, an optimal route shall pass by a sequence of locations in the city map labeled by these keywords. At the same time, this route can be subject to some constrains like travel time, money and so on.

This type of travel route search problem has been substantially studied. As two examples, the weighted constraint shortest path problem (WCSPP) [Dumitrescu and Boland, 2003] and the shortest path problem with time windows (SPPTW) [Desrochers and Soumis, 1988] aim to find a route with the shortest travel distance under a certain threshold (such as the travel time). However, they do not consider user preferences and thus are not able to satisfy specific requirements of the user. Some studies take into account user preferences, such as the TPQ [Li *et al.*, 2005] and the OSR [Sharifzadeh *et al.*, 2008] query. These methods retrieve routes that pass by all the user-specified types of locations with short travel distances. However, a user may not be satisfied with the proposed routes due to some constraints, e.g., they may not have enough holiday time to follow the proposed route. The more recent work [Cao *et al.*, 2012] finds an optimal route that passes by user-specified types of locations and satisfies a certain budget constraint. It assumes that if a location is labeled by a keyword that expresses a user's preference, the location can fully satisfy the user's need; however, this may not always be true.

As identified in previous research, a location can have multiple functionalities and can be labeled by multiple keywords (such as "restaurant" or "mall") by different users. Fig. 1 shows a road network where each location is labeled by some keywords, each of which is associated with the degree to which the location satisfies the user's need expressed by this keyword (such as mall). The associated $[0, 1]$ weighting reflects the popularity of the location as a given keyword, and the degree to which it satisfies user preferences.

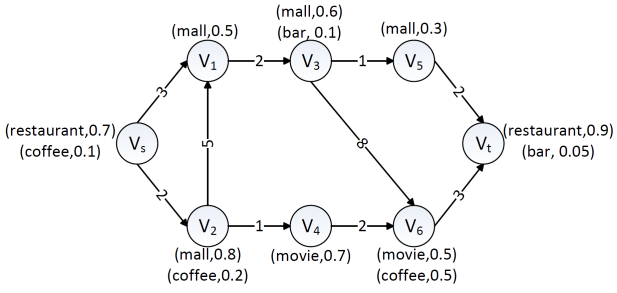


Figure 1: An example road network with locations labeled by multiple keywords.

In this paper, we allow a user to weight their various preferences, and we assume that each location is only able to satisfy their preferences to a certain degree. Our objective is to find a route passing by several locations that can optimally satisfy the user's weighted preferences. However, how to measure the degree to which a route covers the keywords indicating the user's preferences is non-trivial. Simply accumulating the keyword degree associated with the locations in a route cannot well reflect the satisfiability of the route. In Fig 1, when a user has a budget score of 8 and weights *mall*, *movie*, *coffee* by 0.5, 0.4, and 0.1 respectively, the route $\langle v_s, v_1, v_3, v_5, v_t\rangle$ is found through an accumulative function of the keyword degrees since it has a budget score 8 and a maximum accumulated weighted degree score $(0.5*(0.6+0.6+0.8)+0.4*0+0.1*0.1 = 1.01)$. However, the route is rather monotonic and concentrates mainly on *mall*.

In our work, we resort to the coverage function [El-Arini *et al.*, 2009] that calculates the *joint* satisfiability of one route over a set of keywords. We reformulate this problem as the *optimal route search for keyword coverage* (ORS-KC), which considers both user preferences and a constraint during the route search, and aims to find a solution that maximizes the keyword coverage function with some budget constraint. Given the example in Figure 1, we would return the route $\langle v_s, v_2, v_4, v_6, v_t\rangle$, which can well satisfy all the user's requirements.

As the coverage function we use is submodular, solving the optimization problem with the constraint is extremely hard particularly for a graph. We prove the ORS-KC problem to be NP-hard by a reduction from the budgeted maximum coverage problem [Khuller *et al.*, 1999]. To avoid the exhaustive search of all routes in a graph, we adapt the A* algorithm [Pearl, 1984] for solving the submodular function maximization problem in a graph. By exploiting the submodular property, we design one *admissible* heuristic function in the search so that the solution optimality is preserved.

In summary, the contribution of this paper is twofold: firstly, we formulate the problem of the optimal route search for keyword coverage (ORS-KC), and prove that this problem is NP-hard; secondly, we propose the A* based search algorithms for solving this problem, and present experiments conducted on two real-world datasets, which demonstrate the efficiency and accuracy of the proposed algorithms.

2 Related Work

Route search and planning is an important problem and has been studied substantially due to its wide range of applications. One of the most classic and well known problem is the shortest path problem, which does not take into account the user preferences and the budget constraint.

Searching for the shortest path under a budget constraint, such as WCSPP [Dumitrescu and Boland, 2003] and SPPTW [Desrochers and Soumis, 1988], is proved to be NP-hard and approximation algorithms are proposed. Similarly, route recommendation (e.g., [Chekuri and Pal, 2005]) finds general popular routes without satisfying users' specific requirements.

Recently, consideration of the user preferences appears in the route search or planning. For example, Li *et al.* [Li *et al.*, 2005] propose the TPQ query, which finds the shortest path between a source and a target location passing by all user-specified types of locations. The work [Sharifzadeh *et al.*, 2008] proposes the OSR query, which finds a shortest route from a specified starting point passing by a sequence of user-specified types of locations. Similar proposals also exist [Chen *et al.*, 2008; Levin *et al.*, 2010; Li *et al.*, 2013]. However, these studies do not take into account a budget constraint.

The work takes into account both the user preferences and the budget constraint in route search [Cao *et al.*, 2012]. However, it treats each keyword equally and assumes that a location can fully satisfy the user's preferences expressed by the keywords labeled with it. In addition, they formulate the objective function as one accumulative function while we use a submodular function, which makes the problem more challenging.

The submodular coverage function we adopt in our work has been used widely in many problems, such as document recommendation [El-Arini *et al.*, 2009], placement of sensor networks [Krause *et al.*, 2011], point-of-interest recommendation [Chen *et al.*, 2015] and so on.

The problem of route search in a graph optimizing a submodular function with a budget constraint has been studied in the work [Chekuri and Pal, 2005], where an recursive greedy algorithm with a performance guarantee is proposed. The algorithm is useful theoretically, but is not applicable to real-world problems such as city route planning. As shown in the study [Singh *et al.*, 2007], its runtime is acceptable only on a graph containing up to 22 nodes, which is not applicable in a road network with at least thousands of locations.

3 Problem Formulation

In this section, we present the keyword coverage function and formally define the problem of *optimal route search for keyword coverage* (ORS-KC). We also prove its complexity.

3.1 Keyword Coverage of a Route

We define a road network as a graph.

Definition 1. Road Network Graph. *A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each node $v \in \mathcal{V}$ represents a location and is associated with a set of keywords denoted by $\mathcal{K}_{v_i} = \{\kappa_{i_1}, \cdots, \kappa_{i_q}\}$, each edge $\langle v_i, v_j\rangle \in \mathcal{E}$ represents a directed route between two locations v_i and v_j in $\mathcal{V}$ and is associated with a cost $b(v_i, v_j)$ (representing travel time, distance etc).*

We only consider directed graphs in this paper; but we note that it is straightforward to extend it to undirected graphs.

Definition 2. Route. *Given a road network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a route $R = \langle v_1, \cdots, v_m\rangle$ is a path from v_1 to v_m in $\mathcal{G}$, i.e., $\forall 1 \leq i \leq m \cdot v_i \in \mathcal{V}$ and $\forall 1 \leq i < m \cdot \langle v_i, v_{i+1}\rangle \in \mathcal{E}$.*

We consider two attributes for a route, namely the *budget* value and the *keyword coverage* value. The *budget* value of a route is a sum-up of the costs of the edges along the route, as shown in Eq. 1 below,

$$BS(R) = \sum_{i=1}^{m-1} b(v_i, v_{i+1}). \quad (1)$$

Given a budget constraint, our goal is to find a route that can best satisfy the user's weighted preferences, where the satisfiability of a route is measured by a *keyword coverage* function [El-Arini *et al.*, 2009]. A route is a traversal of locations each of which is associated with multiple keywords, and the *keyword coverage* function reflects the degree to which a set of query keywords are covered by the route.

Let $\mathcal{K} = \{\kappa_1, \cdots, \kappa_q\}$ be a set of query keywords and $\lambda_{\kappa_q}(>0)$ be the weight of keyword κ_q. The coverage function we use to compute the keyword coverage for a route $R = \langle v_1, \cdots, v_m\rangle$ is shown in Eq. 2,

$$KC(R) = \textstyle\sum_{\kappa_q \in \mathcal{K}} \lambda_{\kappa_q} cov_{\kappa_q}(R), \quad (2)$$

where $cov_{\kappa_q}(R)$ measures the degree to which the keyword κ_q is covered by at least one location in R as computed below:

$$cov_{\kappa_q}(R) = 1 - \textstyle\prod_{v_i \in R}[1 - cov_{\kappa_q}(v_i)], \quad (3)$$

where $cov_{\kappa_q}(v_i)$ is the degree to which the location v_i covers the keyword κ_q.

For example, given the aforementioned case in Fig. 1, the keyword coverage of the route can be calculated as: $KC(\langle v_s, v_2, v_4, v_6, v_t\rangle) = 0.5*[1-(1-0.7)]+0.4*[1-(1-0.5)*(1-0.2)]+0.1*[1-(1-0.1)*(1-0.1)*(1-0.5)] = 0.6495$.

Note that the keyword coverage function $KC(R)$ is submodular so that the property is satisfied, e.g., $KC(R_1 \bigcup v_i) - KC(R_1) \geq KC(R_2 \bigcup v_i) - KC(R_2)$, for all $R_1 \subseteq R_2 \subseteq \mathcal{R}$ and $v_i \notin R_1$, where $\mathcal{R}$ is a finite set of routes and $R_1(R_2) \bigcup v_i$ composes a new route.

In this paper, we use the popularity of the location v_i labeled by the keyword κ_q to measure $cov_{\kappa_q}(v_i)$, which is implied by the number of check-ins that the location receives at v_i labeled by κ_q. To equally prioritize locations with a high volume of check-ins, we make $cov_{\kappa_q}(v_i)$ proportional to the average users' check-ins at v_i labeled by κ_q, and is computed in Eq. 4,

$$cov_{\kappa_q}(v_i) = min[\frac{nc_{v_i}^{\kappa_q}}{\frac{1}{\sum_{v_i} 1} \times \sum_{v_i} nc_{v_i}^{\kappa_q}}, 1], \quad (4)$$

where $nc_{v_i}^{\kappa_q}$ is the number of check-ins that are made at v_i and labeled by keyword κ_q, $\sum_{v_i} 1$ is the number of locations visited, and $\sum_{v_i} nc_{v_i}^{\kappa_q}$ counts all check-ins labeled by κ_q. $cov_{\kappa_q}(v_i)$ is 1 if $nc_{v_i}^{v_q}$ exceeds the average number of check-ins labeled by κ_q.

3.2 Keyword Coverage Optimal Route Search

Intuitively, the *optimal route search for keyword coverage* (ORS-KC) problem consists in finding an optimal route from a source node to a target in a graph, such that the keyword coverage is optimized and the budget score satisfies a given constraint. Formally, we define the ORS-KC problem as follows: given $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and a query $\mathcal{Q} = \langle v_s, v_t, \mathcal{K}, \Upsilon, \Delta\rangle$, where v_s is the source location, v_t is the target location, $\mathcal{K}$ is a set of keywords, Υ contains the keyword weights λ_{κ_q}, and Δ specifies a budget constraint, we aim to find the route R starting from v_s and ending at v_t such that

$$\begin{array}{l} R = argmax_R \ KC(R) \\ subject\ to \ BS(R) \leq \Delta. \end{array} \quad (5)$$

Proposition 1. *Solving the optimal route search for keyword coverage (in Eq. 5) is an NP-hard problem.*

Proof. We develop the proof by reducing the problem from a unit cost version of the budgeted maximum coverage (UBMC) problem [Khuller *et al.*, 1999]. Given a collection of sets $S = \{S_1, S_2, \cdots, S_m\}$ with a unit cost C, a domain of elements $X = \{x_1, x_2, \cdots, x_n\}$ with associated weights $\{w_1, w_2, \cdots, w_n\}$, and a budget Δ, the aim of UBMC is to find a collection of sets $S' \subseteq S$ whose total budget is smaller than L and the elements covered by S' have the largest total weight.

Given an instance of the UBMC problem φ, we can construct an ORS-KC problem instance ω as follows: we build a graph containing $m + 2$ nodes, where two nodes are the source node and the target node and the other nodes correspond to the sets in S, and each pair of nodes is connected by an edge with a unit cost C. For each element x_j, we create a query keyword κ_j, and the weight of κ_j is set to the weight w_j of x_j. On a node v_i corresponding to a set S_i, for each element $x_j \in S_i$ we associate κ_j with v_i, and the value $cov_{\kappa_j}(v_i)$ is set to 1. The budget value Δ is set to $(L + C)$. Given this mapping, if S' is the optimal result of φ, any route passing by the nodes corresponding to the sets in S' is the optimal route of ω, and vice versa. As the UBMC problem has been proved to be NP-hard, the ORS-KC problem is NP-hard as well. ■

In analogy to the *submodular orienteering problem* [Chekuri and Pal, 2005], the ORS-KC problem is hard to solve in an optimal way. One potential approximation with theoretical quality bound is $\frac{1}{\lceil 1+log(k)\rceil}$, where k is the number of nodes in the optimal route. As demonstrated in the work [Singh *et al.*, 2007], the approximate technique costs more than 10^4 seconds in a small graph with 22 nodes and a budget of 450 meters, so it is not scalable and cannot be used to solve the problem in real road graphs which typically have thousands of nodes and much larger budgets. The problem is difficult because the route must satisfy the budget constraint and must have an optimized keyword coverage score, which is computed by a submodular function.

4 A* Search Algorithm

We propose an adaptation of the A* algorithm for solving the ORS-KC problem. We first introduce the pre-processing method and then present the algorithm. Meanwhile we develop a pruning techniques to improve the search efficiency.

4.1 Pre-processing

We introduce the pre-processing method that is commonly used to accelerate the search algorithm in a road network. We use the *Floyd-Warshall* algorithm [Floyd, 1962], which is a well-known algorithm for finding all pairs of the shortest

paths, to compute the smallest budget for each pair of locations. After this pre-processing, we get all the smallest budgets of pairs. For every pair of nodes (v_i, v_j), we denote the least budget from v_i to v_j as $BS_{sm}(v_i, v_j)$.

4.2 Algorithm

A brute-force approach for solving ORS-KC is to conduct an exhaustive search. It first enumerates all candidate paths from a source node and uses a queue to store the partial paths. Subsequently, in each step, it extends one partial path in the queue and generates a new set of candidate partial paths. The paths that have budget scores smaller than Δ are added to the queue. Finally it returns the best route after comparing all the candidate routes from the source location to the target location. Thus *the brute-force technique guarantees an optimal solution to the ORS-KC problem*.

However, the exhaustive search is computationally prohibitive. To avoid enumerating all partial paths, we propose a novel variant of the A* algorithm. The basic idea of applying the A* algorithm is to search the candidate partial paths with best estimated keyword coverage firstly, and prune the partial paths with small estimated keyword coverage. It is non-trivial to estimate the keyword coverage of partial paths because the objective function is submodular and the budget also needs to be considered.

The Framework of A* Algorithm

As all the partial paths start from a source node, we can build a search tree and conduct a breadth-first search. As well known, the A* algorithm uses a knowledge-plus-heuristic cost function to determine the order in which the search visits nodes in the tree. When we reach a node v_n in the search tree, we define the knowledge-plus-heuristic cost function at node v_n by:

$$f^n(R_{s\to t}) = g^n(R_{s\to n}) + h^n(R_{n\to t}|R_{s\to n}), \qquad (6)$$

where $g^n(R_{s\to n})$ is the *exact* keyword coverage of the path $R_{s\to n}$ and can be computed using $KC(R_{s\to n})$ in Eq. 2, and $h^n(R_{n\to t}|R_{s\to n})$ estimates the *marginal* keyword coverage of the successive route conditioned on $R_{s\to n}$ in the search tree. Note that $h^n(\cdot)$ does not depend on the $g^n(\cdot)$ in the conventional A* algorithm, and thus designing $f^n(R_{s\to t})$ in our problem is more challenging.

From the source node v_s to the node v_i, there exist many paths. To store each path and its information on the node v_i, we define the **route label** in format of $L_i^k = \langle R_i^k, BS(R_i^k), KC(R_i^k), f^i(R_{s\to t})\rangle$, where R_i^k represent the k^{th} path from v_s to v_i. We use a max-priority queue Q to organize these labels, which are enqueued into Q in decreasing order of $f^i(R_{s\to t})$.

In Alg. 1, the algorithm starts by creating an empty R that is used to store the current best route (line 1). The current best keyword coverage score is stored in KC_{max} and the route label containing v_s is enqueued into Q (lines 3-4). It dequeues the candidate route label from Q one by one until either Q is empty or all the route labels in Q have an estimated keyword coverage smaller than KC_{max} (lines 5-7). In each while-loop, the algorithm first obtains a partial route from the dequeued route label that is to be extended (line 8). For each outgoing neighbor v_j of v_i, it creates a new route (line 10) and ignores the new route whose budget score is larger than the budget Δ (line 11). The algorithm updates R and KC_{max} when the keyword coverage of a new route is larger than the current KC_{max}; otherwise, a candidate partial route is found and its keyword coverage is estimated. For the candidate route, the algorithm then creates a new route label and enqueues it into Q when the estimated keyword coverage is larger than the current KC_{max} (lines 12-21). Finally, if KC_{max} is never updated, there exists no feasible route for the given query; otherwise, the optimal route R is returned (lines 22-23).

Algorithm 1: A* Algorithm for ORS-KC

Input: $\mathcal{G} = (V, E)$, $\mathcal{Q} = \langle v_s, v_t, \mathcal{K}, \Upsilon, \Delta\rangle$, $BS_{sm}(v_i, v_j)$ of all pairs of locations in $\mathcal{G}$
Output: An optimal route R

1 Initialize a max-priority queue $Q \leftarrow \emptyset$;
2 $R \leftarrow \emptyset$; $KC_{max} \leftarrow -\infty$;
3 Create a route label: $L_s^0 \leftarrow \langle (v_s), 0, 0, 0\rangle$;
4 Q.enqueue(L_s^0);
5 **while** Q *is not empty* **do**
6 $L_i^k \leftarrow Q$.dequeue();
7 **If** $L_i^k.f^n(R_{s\to t}) \leqslant KC_{max}$ **break**;
8 Obtain R_i^k from L_i^k;
9 **for** *each edge* (v_i, v_j) **do**
10 Create a route $R_j^l \leftarrow R_i^k \cup v_j$;
11 **If** $BS(R_j^l) > \Delta$ **continue**;
12 **if** v_j *is* v_t **then**
13 **if** $KC(R_j^l) > KC_{max}$ **then**
14 $R \leftarrow R_j^l$;
15 $KC_{max} \leftarrow KC(R_j^l)$;
16 **else**
17 Compute $f^j(R_{s\to t})$;
18 **if** $f^j(R_{s\to t}) > KC_{max}$ **then**
19 Create a route label $L_j^l \leftarrow$
20 $\langle R_j^l, BS(R_j^l), KC(R_j^l), f^j(R_{s\to t})\rangle$;
21 Q.enqueue(L_j^l);
22 **If** KC_{max} is $-\infty$ **return** "No feasible route exists";
23 **Else return** R;

The key challenge of this algorithm is at line 17, i.e., how to compute the function in Eq. 6. Because the keyword coverage function is a submodular function, it implies that the future path-keyword coverage $h^n(\cdot)$ depends on the past path-keyword coverage $g^n(\cdot)$, which is different from the conventional A* algorithm. We next proceed to present how to compute the function $f^n(\cdot)$.

The Heuristic Function $h^n(\cdot)$

The most crucial part of designing an A* algorithm is to compute the *admissible heuristic function* for a node v_n, i.e., in our problem it is $h^n(\cdot)$. In conventional A* algorithms, it is the estimation of the cost value of the route from v_n to the target node v_t. However, because our key-

word coverage function $KC(\cdot)$ is submodular, as shown in Eq. 6, the heuristic function $h^n(\cdot)$ is the *marginal* keyword coverage of the successive route conditioned on $R_{s\to n}$, instead of the keyword coverage of the route $R_{n\to t}$. Therefore, rather than estimating $KC(R_{n\to t})$, we are going to estimate $KC(R_{s\to t}) - KC(R_{s\to n})$.

Recall that $KC(R_{s\to t}) = \sum_{\kappa_q\in\mathcal{K}} \lambda_{\kappa_q}(1 - \prod_{v_i\in R_{s\to t}}[1 - cov_{\kappa_q}(v_i)])$, and $KC(R_{s\to n}) = \sum_{\kappa_q\in\mathcal{K}} \lambda_{\kappa_q}(1 - \prod_{v_i\in R_{s\to n}}[1 - cov_{\kappa_q}(v_i)])$. Hence, we have:

$$
\begin{aligned}
&KC(R_{s\to t}) - KC(R_{s\to n})\\
&= \sum_{\kappa_q\in\mathcal{K}} \lambda_{\kappa_q}\Big(\prod_{v_i\in R_{s\to n}}[1 - cov_{\kappa_q}(v_i)] - \prod_{v_j\in R_{s\to t}}[1 - cov_{\kappa_q}(v_i)]\Big)\\
&= \sum_{\kappa_q\in\mathcal{K}} \lambda_{\kappa_q}\Big(\prod_{v_i\in R_{s\to n}}[1 - cov_{\kappa_q}(v_i)](1 - \prod_{v_i\in R_{n\to t}}[1 - cov_{\kappa_q}(v_i)])\Big)\\
&\le \max_{\kappa_q\in\mathcal{K}} \prod_{v_i\in R_{s\to n}}[1 - cov_{\kappa_q}(v_i)] \sum_{\kappa_q\in\mathcal{K}} \lambda_{\kappa_q}(1 - \prod_{v_i\in R_{n\to t}}[1 - cov_{\kappa_q}(v_i)]).
\end{aligned}
$$

Note that $KC(R_{n\to t}) = \sum_{\kappa_q\in\mathcal{K}} \lambda_{\kappa_q}(1 - \prod_{v_i\in R_{n\to t}}[1 - cov_{\kappa_q}(v_i)])$, and thus we can conclude:

$$
\begin{aligned}
&KC(R_{s\to t}) - KC(R_{s\to n}) \le \max_{\kappa_q\in\mathcal{K}} \prod_{v_i\in R_{s\to n}}[1 - cov_{\kappa_q}(v_i)]) * \\
&KC(R_{n\to t}).
\end{aligned}
$$

Hence, we can define $h^n(R_{n\to t}|R_{s\to n}) = (\max_{\kappa_q\in\mathcal{K}} \prod_{v_i\in R_{s\to n}}[1 - cov_{\kappa_q}(v_i)]) \cdot KC(R_{n\to t})$, and this estimation offers the expected *admissibility* of the heuristic function, as it is an upper bound of $KC(R_{s\to t}) - KC(R_{s\to n})$. This guarantees the optimality of the solution.

Specifically, when reaching a node v_n, we first get $(\max_{\kappa_q\in\mathcal{K}} \prod_{v_i\in R_{s\to n}}[1 - cov_{\kappa_q}(v_i)])$, which is already known, and then we compute $KC(R_{n\to t})$. Computing $KC(R_{n\to t})$ becomes a new ORS-KC problem with the reduced budget $\Delta' = \Delta - BS_{sm}(v_s, v_n)$. Fortunately, it is not necessary to compute the exact $KC(R_{n\to t})$, and instead we can estimate an upper bound (UB) for $KC(R_{n\to t})$. Then, we compute $h^n(R_{n\to t}|R_{s\to n})$ as $(\max_{\kappa_q\in\mathcal{K}} \prod_{v_i\in R_{s\to n}}[1 - cov_{\kappa_q}(v_i)]) \cdot UB$, which still guarantees admissibility. We proceed to explain how to estimate the upper bound of $KC(R_{n\to t})$.

We first get the locations set L_n including all nodes which can be visited from v_n to v_t with the budget Δ', denoted by $L_n = \{v_i | BS_{sm}(v_n, v_i) + BS_{sm}(v_i, v_t) \le \Delta'\}$. We compute the node's cost as a lower bound of the budget passing by this node, i.e., $Cost(v_i) = \frac{min\{b(v_k,v_i)\}+min\{b(v_i,v_j)\}}{2}$, where $\langle v_k, v_i\rangle$ is one in-edge of v_i and $\langle v_i, v_j\rangle$ is one out-edge of v_i. We set $Cost(v_t) = 0.0$ at the target node. Then, we utilize the greedy algorithm for the budgeted maximum coverage (BMC) problem [Khuller *et al.*, 1999], to approximately find a set of nodes such that their total cost does not exceed the budget Δ', and their keyword coverage is maximized. Note that using this algorithm, the nodes found are not necessarily connected as a route, but this does not affect the fact that the result found provides an upper bound. This algorithm achieves an approximation factor of $1-1/\sqrt{e}$. Thus, we can get $KC(L^{op}) \leqslant \frac{KC(L^{mg})}{1-1/\sqrt{e}}$, where L^{mg} is the nodes set found by the greedy algorithm and L^{op} is the optimal nodes set. Proposition 2 provides an upper bound of $KC(R_{n\to t})$.

Proposition 2. $KC(R_{n\to t}) \le \frac{KC(L^{mg})}{1-1/\sqrt{e}}$.

Proof. We first prove that the optimal route R^{op} achieved by computing $KC(R_{n\to t})$ is a feasible solution of the transformed BMC problem. For the optimal route R^{op}, the budget score $BS(R^{op})$ is not larger than Δ', e.g., $\Delta' \ge BS(R^{op})$. As $BS(R^{op}) = \sum_{(v_i,v_j)\in R^{op}} b(v_i, v_j)$ is larger than $\sum_{v_i\in R^{op}} Cost(v_i)$, we have $\sum_{v_i\in R^{op}} Cost(v_i) \le BS(R^{op}) \le \Delta'$, and thus R^{op} is a feasible solution. Since L^{op} is the optimal solution, we know that $KC(R^{op}) \le KC(L^{op}) \leqslant \frac{KC(L^{mg})}{1-1/\sqrt{e}}$. ■

In summary, we set $h^n(R_{n\to t}|R_{s\to n}) = (\max_{\kappa_q\in\mathcal{K}} \prod_{v_i\in R_{s\to n}}[1 - cov_{\kappa_q}(v_i)]) \frac{KC(L^{mg})}{1-1/\sqrt{e}}$, which preserves the admissibility of the heuristic function. We finally compute the function $f^n(R_{s\to t})$ in Eq. 7,

$$
\begin{aligned}
&f^n(R_{s\to t}) = KC(R_{s\to n}) +\\
&(\max_{\kappa_q\in\mathcal{K}} \prod_{v_i\in R_{s\to n}}[1 - cov_{\kappa_q}(v_i)]) \frac{KC(L^{mg})}{1-1/\sqrt{e}}.
\end{aligned} \tag{7}
$$

Pruning Optimization

In Alg.1, we only prune the candidate partial routes with small keyword coverage (lines 18-21). We further improve the efficiency of the A* algorithm by pruning the candidate partial routes with large budget scores. Before creating a new route label for R_j^l and enqueuing it into Q, we check whether the estimated budget score $BS(R_j^l) + BS_{sm}(v_j, v_t)$ is larger than the budget Δ. If $BS(R_j^l) + BS_{sm}(v_j, v_t) > \Delta$, we prune R_j^l. Proposition 3 states that the pruning technique preserves all feasible solutions.

Proposition 3. *No feasible route contains R_j^l given $BS(R_j^l) + BS_{sm}(v_j, v_t) > \Delta$.*

Proof. Assume that when $BS(R_j^l) + BS_{sm}(v_j, v_t) > \Delta$, there exists a feasible route $R_{s\to j\to t}^l$ including R_j^l, where $R_{s\to j\to t}^l = R_j^l + R_{j\to t}^l$ and $BS(R_{s\to j\to t}^l) \le \Delta$, so $BS(R_{s\to j\to t}^l) = BS(R_j^l) + BS(R_{j\to t}^l) \le \Delta$. Consequently, we get $BS(R_{j\to t}^l) < BS_{sm}(v_j, v_t)$, which contradicts the assumption. ■

5 Experimental Study

We conducted a series of experiments to study the ORS-KC problem and demonstrate the algorithm performance.

5.1 Experimental Settings

Datasets. We use two real-world datasets. One has been collected from $Foursquare$ which was made in Singapore (SG) between Aug. 2010 and Jul. 2011 [Yuan *et al.*, 2013], and another one is from $Gowalla$ which was made in Austin (AS) between Nov. 2009 and Oct. 2010 [Cho *et al.*, 2011]. We have adopted the $Foursquare$ APIs to fill in the missing values of keywords (categories of locations). The SG dataset

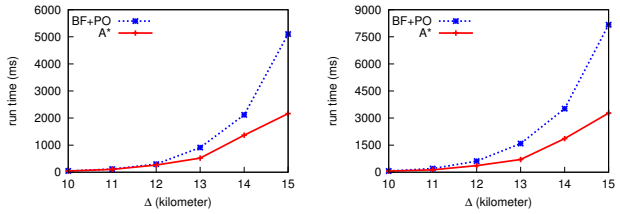
has 189,306 check-ins made by 2,321 users at 5,412 locations, and the AS dataset contains 201,525 check-ins made by 4,630 users at 6,176 locations. Following the work [Cao *et al.*, 2012], we build an edge between two locations which were visited continuously in 1 day by the same user, and set the budget value by the Euclidean distance for each edge.

Comparative Methods. We have implemented the A* algorithm with the above described pruning optimization. In addition, we adopt the standard Weighted A* method (WA*) [Ebendt and Drechsler, 2009] to further optimize performance. In WA*, the cost function for WA* is expressed as $f^n(\cdot) = g^n(\cdot) + \omega h^n(\cdot)$, which improves performance through a phenomenon known as precision-complexity exchange, speeding up search at the cost of a minimal departure from optimality [1]. For the comparison purpose, we have implemented the Brute-Force (BF) approach. Meanwhile we use the Pruning Optimization (as described in Section 4) to improve BF (denoted by BF+PO).

We generate 100 queries randomly in each experiment, implement methods in JAVA and conduct experiments on a Windows PC with a 4-core Intel i7-870 2.93GHz CPU and 16 GB memory.

5.2 Performance of Methods

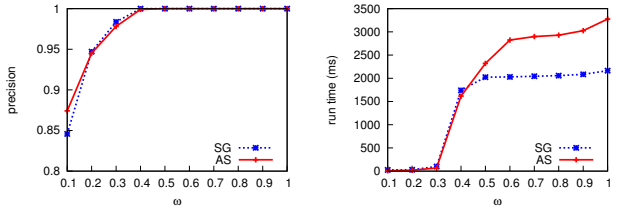
Efficiency of Methods. We compare our A* algorithm with BF+OP method on the run time with variation of the budget limit Δ (travel distance). Note that both methods are able to get the optimal solution, and BF+OP is the improved version of brute-force approach. Fig. 2 shows that the A* algorithm is usually $2-3$ times faster than BF+OP method, and the run time of both methods grows with the increase of Δ quickly, as Δ is larger, more potential routes should be checked.



(a) SG (b) AS

Figure 2: Comparison of methods varying Δ on run time.

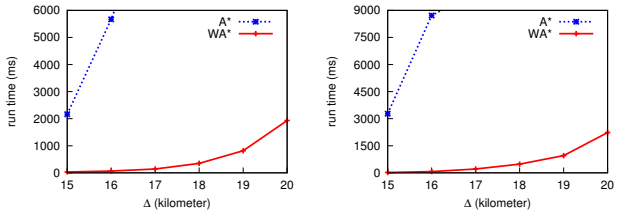
Effect of the Weighting Parameter ω. In the WA* algorithm, the weighting parameter ω sets a trade-off between precision and complexity. We denoted the route found by the WA* algorithm as R^{sw}, if $\omega = 1$, the WA* algorithm gets the optimal route R^{op}, then we use the ratio $\frac{KC(R^{sw})}{KC(R^{op})}$ to measure the precision of the WA* algorithm. Fig. 3 shows the effect of ω with $\Delta = 15\ kilometers$ on solving the ORS-KC problem. With the increase of ω, the precision improves while the efficiency drops (longer run time). In particular, the change of precision and run time is notable from $\omega = 0.3$ to $\omega = 0.4$, which indicates that $0.4h^n(\cdot) = 0.4 * \frac{KC(L^{mg})}{1-1/\sqrt{e}}$ is close to the upper bound of the marginal keyword coverage $h^n(R_{n\to t}|R_{s\to n})$. As $(1-1/\sqrt{e})$ approximates 0.4, $KC(L^{mg})$ is near the upper bound of $h^n(R_{n\to t}|R_{s\to n})$ in most cases. We thus set the weighting coefficient to 0.2, which drastically reduces computation time while limiting solution quality deterioration to 5%.



(a) Effect on the Precision (b) Effect on the Efficiency

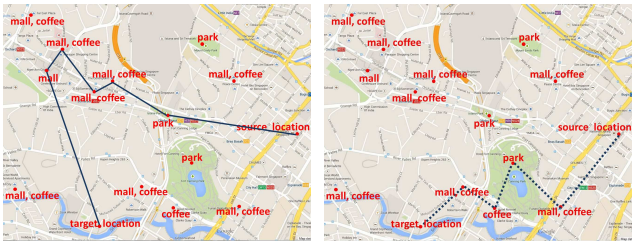
Figure 3: Effect of weighting parameter ω with $\Delta = 15\ kilometers$ in the WA* algorithm.

Scalability of the A* algorithm. In order to study the scalability of the algorithms, we run the A* and WA* algorithms on a larger budget limit Δ in two datasets. As the run time of A* algorithm is larger than $10^4\ ms$ when $\Delta \geq 17\ kilometers$, we omit it. Fig. 4 shows that WA* scales better than A*. This is expected as a large budget enlarges A*'s search space.



(a) SG (b) AS

Figure 4: Scalability of the algorithms varying Δ.



(a) Route1 (b) Route2

Figure 5: Routes selected by (W)A* for solving ORS-KC problems in *Singapore*.

Example. We use one real-world example in the SG dataset to show that ORS-KC can find optimal routes for satisfying user's various preferences. We set the source location at *National Library of Singapore* and the target location at *Gallery*

[1] As discussed in the work [Ebendt and Drechsler, 2009] weighted A* is a bounded suboptimality method. Another way to characterize the precision-complexity exchange is to consider the approach as epsilon-admissible.

Hotel, the budget constraint $\Delta = 6\ kilometer$, and the keywords set is $\mathcal{K} = \{\kappa_1 = mall, \kappa_2 = coffee, \kappa_3 = park\}$, i.e., a user would like to enjoy shopping, drink some coffee and visit a park on the route. To keep the example clear, we only list some locations having the three keywords in the map. When we set the keyword weights be $\Upsilon = \{\lambda_{\kappa_1} = 0.8, \lambda_{\kappa_2} = 0.1, \lambda_{\kappa_3} = 0.1\}$, our (W)A* algorithm returned the optimal route in Fig.5(a), which contains some popular malls on the prosperous *Orchard Road*. On the other hand, when $\Upsilon = \{\lambda_{\kappa_1} = 0.1, \lambda_{\kappa_2} = 0.1, \lambda_{\kappa_3} = 0.8\}$, the route in Fig. 5(a) is not a good choice, because it only goes though a small park. Instead the (W)A* algorithm selects a new route that contains a popular park in Fig. 5(b).

6 Conclusion

Considering users' various preference on route search, we introduce the keyword coverage function and define the *optimal route search for keyword coverage* (ORS-KC) problem, which is to find an optimal route such that it can optimally satisfy the user's weighted preferences. In order to solve ORS-KC, we define an admissible heuristic exploiting the submodular property, then use a variant of A* to compute solution routes. This is also challenging in a general submodular function optimization for a graph. Empirical results demonstrate performance of our methods as well as the quality of routes found in the ORS-KC problem. In the future work, we will seek other approximate algorithms for solving this new problem.

Acknowledgements

This work was done during Xuefeng Chen's visit to Teesside University, sponsored by Teesside University. Yanping Xiang would like to thank the support of projects (No. 2015TD0002 and No. 2014FZ0087) from the Sichuan province, China.

References

[Cao *et al.*, 2012] Xin Cao, Lisi Chen, Gao Cong, and Xiaokui Xiao. Keyword-aware optimal route search. *PVLDB*, 5(11):1136–1147, 2012.

[Chekuri and Pal, 2005] Chandra Chekuri and Martin Pal. A recursive greedy algorithm for walks in directed graphs. In *FOCS*, pages 245–253, 2005.

[Chen *et al.*, 2008] Haiquan Chen, Wei-Shinn Ku, Min-Te Sun, and Roger Zimmermann. The multi-rule partial sequenced route query. In *GIS*, pages 1–10, 2008.

[Chen *et al.*, 2015] Xuefeng Chen, Yifeng Zeng, Gao Cong, Shengchao Qin, Yanping Xiang, and Yuanshun Dai. On information coverage for location category based point-of-interest recommendation. In *AAAI*, pages 37–43, 2015.

[Cho *et al.*, 2011] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *KDD*, pages 1082–1090, 2011.

[Desrochers and Soumis, 1988] M. Desrochers and F. Soumis. A generalized permanent labeling algorithm for the shortest path problem with time windows. *Information Systems Research*, 26(1):191–212, 1988.

[Dumitrescu and Boland, 2003] I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003.

[Ebendt and Drechsler, 2009] Rüdiger Ebendt and Rolf Drechsler. Weighted A* search–unifying view and application. *Artificial Intelligence*, 173(14):1310–1342, 2009.

[El-Arini *et al.*, 2009] Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In *KDD*, pages 289–298, 2009.

[Floyd, 1962] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[Khuller *et al.*, 1999] Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[Krause *et al.*, 2011] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. Robust sensor placements at informative and communication-efficient locations. *ACM Transactions on Sensor Networks (TOSN)*, 7(4):1–33, 2011.

[Levin *et al.*, 2010] Roy Levin, Yaron Kanza, Eliyahu Safra, and Yehoshua Sagiv. Interactive route search in the presence of order constraints. *PVLDB*, 3(1):117–128, 2010.

[Li *et al.*, 2005] Feifei Li, Dihan Cheng, Marios Hadjieleftheriou, George Kollios, and Shang-Hua Teng. On trip planning queries in spatial databases. In *SSTD*, pages 273–290, 2005.

[Li *et al.*, 2013] Jing Li, Yin David Yang, and Nikos Mamoulis. Optimal route queries with arbitrary order constraints. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 25(5):1097–1110, 2013.

[Pearl, 1984] Judea Pearl. Heuristics: intelligent search strategies for computer problem solving. *Addison-Wesley*, 1984.

[Sharifzadeh *et al.*, 2008] Mehdi Sharifzadeh, Mohammad R. Kolahdouzan, and Cyrus Shahabi. The optimal sequenced route query. *VLDB Journal*, 17(4):765–787, 2008.

[Singh *et al.*, 2007] Amarjeet Singh, Andreas Krause, Carlos Guestrin, William J Kaiser, and Maxim A Batalin. Efficient planning of informative paths for multiple robots. In *IJCAI*, pages 2204–2211, 2007.

[Yuan *et al.*, 2013] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. Time-aware point-of-interest recommendation. In *SIGIR*, pages 363–372, 2013.