Multi-user Routing to Single Destination with Confluence

Kazuki Takise, Yasuhito Asano and Masatoshi Yoshikawa
Kyoto University
Kyoto, Japan 606-8501
takise@db.soc.i.kyoto-u.ac.jp, {asano, yoshikawa}@i.kyoto-u.ac.jp

ABSTRACT

The recent increase in attention to ride-sharing applications demonstrates the importance of routing algorithms for multiple users who obtain benefits from confluence, that is, traveling together on all or part of their routes. We propose novel and flexible formulation of routing optimization for multiple users who have their respective sources and a single common destination. The formulation is general enough to express each user's benefit (or cost) of confluence for every combination of users. Hence, the formulation can represent a wide range of applications and subsumes almost all formulations proposed in literature. We establish an efficient exact method for the formulation. Interestingly, we found well-known Dreyfus-Wagner Algorithm for the Minimum Steiner Tree Problem (MSTP) is extensible for ours, although our formulation is much harder than the MSTP. Our experimental results obtained on large-scale road networks reveal that our method is efficient in practical settings.

Keywords

Confluence; graph; optimization; road network; routing

Categories and Subject Descriptors

G.2 [**DISCRETE MATHEMATICS**]: Graph TheoryGraph algorithms

1. INTRODUCTION

Along with recent attention to ride-sharing applications and research into automatic driving, routing optimization for multiple users who obtain benefits by traveling together is increasing its importance. Routing optimization with confluence can be applied to various applications in which people and objects do not just move by car, but also move in various ways; thus, flexible formulations and efficient optimization methods are essential.

In this paper, we propose a novel and flexible formulation and an efficient algorithm of routing optimization for multiple users who have their respective sources and a single common destination. Our formulation is highly generalized so that it is applicable not only to ride-sharing applications, but also to other various ones in our daily lives. As a simple example, let us consider the situation wherein several friends gather at the venue of an event. Figure 1 illustrates two variants of the routes of three users u_1, u_2 and u_3 to the venue v_T on a road network. Dashed arrows depict their shortest paths to the destination. On the other hand, Solid arrows depict an example of the routes with confluence. Although these routes are longer than the respective shortest paths, the psychological fatigue each user actually suffers is considered to be smaller when traveling together with friends and having conversations.

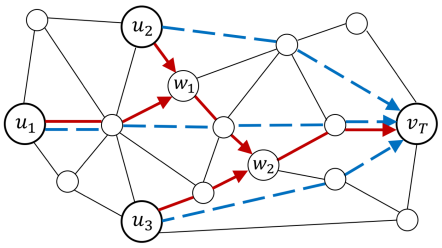


Figure 1: Routes for three users on a road network. Dashed arrows indicate their shortest paths. Solid lines indicate routes for travelling together.

Our formulation of routing optimization for multiple users does not simply use the length of each path itself; rather, we consider that the cost of users traveling together can be discounted, and the total cost of users traveling to the destination might be less than that of the shortest paths. Thus, the routes with confluence, not the shortest paths, can be found as the optimum solution of our formulation. This cost depends on with whom each user travels along the path in many cases. Note that 'traveling cost' does not necessarily represent financial expense, but it might also be the physical or psychological cost of traveling through the path.

Let us consider another application, where a large-scale event such as a sports game or an international conference is held. In this case, users who travel to the venue of the event by taxi can reduce the total travel expense by sharing their routes and using fewer taxis.

In many applications including the above scenarios, the traveling costs vary depending on the number of users traveling together. These cases frequently appear in our daily

SIGSPATIAL'16 October 31 - November 03, 2016, Burlingame, CA, USA

ACM ISBN 978-1-4503-4589-7/16/10.
DOI: http://dx.doi.org/10.1145/2996913.2997018

lives; however, little work has been directed at them. In addition, the traveling costs can also vary depending on the combination of users traveling together. In the case of familiar friends, which we mentioned above, the traveling costs may vary depending on how familiar each pair of friends are.

Moreover, even if some people travel together in the same group, the confluence benefits can vary from user to user. For example, in the situation where elderly people or people with disabilities travel with able-bodied people, the benefit for those elderly people or people with disabilities may be large, while the benefit for able-bodied people is small.

As mentioned above, routing optimization with confluence has various applications. However, no previous research can handle these applications uniformly and efficiently. In this paper, we formulate the above problem and propose an exact optimization method for the formulation. Our experimental results obtained on large-scale road networks reveal that the proposed method is practically efficient.

In this paper, we have two noticable contributions. Firstly, our proposed formulation is generalized so that confluence benefit of each user depends on the members of the group traveling together. This aspect enables our formulation to be applicable to a wide variety of applications. Secondly, we clarify the theoretical relation between two well-known combinatorial optimization problems, the Minimum Steiner Tree Problem (MSTP) and the Buy-at-Bulk Problem, which have many similar aspects, but have not been associated with each other. We give more detailed explanation about these problems in Section 1.1. In fact, we formulate our problem by successfully extracting from these problems several aspects, which are necessary for routing optimization for multiple users. The proposed method is generalized so successfully that it can be executed as efficiently as Dreyfus-Wagner Algorithm, one of the most efficent exact methods for the MSTP. Some approximation methods for the Buy-at-Bulk Problem can be applied to our formulation with an additional constraint.

2. RELATED WORK

2.1 Minimum Steiner Tree Problem

Given an undirected graph and a set of vertices as terminal points, the Minimum Steiner Tree Problem (MSTP) finds a tree interconnecting the terminal points such that the total weight of the edges of the tree is minimized. This problem has been under investigation in various formulations for many years[1][2].

Our problem generalizes the MSTP. Since the MSTP is known to be NP-hard, our problem is also NP-hard. Furthermore, our exact method is a natural, but non-trivial generalization of Dreyfus-Wagner Algorithm[3], which is a well-known exact method for the MSTP. We discover an interesting fact that our exact method is executable as efficiently as Dreyfus-Wagner Algorithm although our problem is a generalization of the MSTP.

2.2 Buy-at-Bulk Problem

The Buy-at-Bulk Problem (BaBP)[4] is one of the network design problems based on a multicommodity network flow problem with demands. The network capacity is bought "wholesale" to guarantee the connectivity from all sources to corresponding sinks. The capacity is sold with a "volume discount": the more capacity that is bought, the lower the price per unit of bandwidth. Various formulations of this problem exist, and the Single-Source Buy-at-Bulk Problem (SSBaBP)[5][6] is especially similar to our problem. Our problem generalizes the cost in the SSBaBP, whereas our problem add a tree constraint of its solution to the SSBaBP. In Section 7, we extend an approximation method for the SSBaBP to our problem. This extension presents a theoretical evidence of the relation between these problems.

2.3 Multi-user Routing

Some research has addressed routing optimization for multiple users considering confluence benefits[7][8]. Our problem is formulated such that the traveling costs can vary depending on the combination of users traveling together unlike the above research.

Some previous research has focused on vehicle ride sharing such as for taxis[9][10][11][12]. In such research, users do not move from their initial points by themselves, and they can only travel to their destinations by car. Thus, in the case of routing optimization to a single destination, our research is more generalized than previous research.

3. PRELIMINARIES

In our problem, we assume that once some users start traveling together, they do not diverge and travel separately. There are two reasons to introduce this assumption. First, it is difficult to estimate the cost of the divergence because the cost depends on various factors in real situations. Rather, in situations where users go to a single destination, users are unlikely to diverge on their paths. Second, this assumption enables our simple formulation and efficient exact method.

3.1 Notation

We model a road network by a directed graph $G(V, E)$, where the vertices V represent locations and the directed edges E represent road segments. Each edge $e \in E$ is given with its length $l_e \in R^+$, which denotes the distance between both ends. Let $d(s, t)$ be the shortest distance between vertices s and t.

Let U be a set of users given in a single query, where each user is given with their initial point $s_u \in V$, and each query also includes a single common destination $v_T \in V$.

Let R_u be a path from s_u to v_T, and $e_u(i)$ represents the i-th edge from s_u in R_u. Let $|R_u|$ be the number of edges in R_u.

3.2 Definition

In this section, we define the concept of confluence, which is one of the most distinguishing aspects of this research.

Definition 1. The **confluence** of two users u_1, u_2 means to start traveling together from a common vertex v to the destination v_T. We call this v a **confluence point** of u_1 and u_2. The confluence point is defined by a pair of integer $i(u_1, u_2)$ and $i(u_2, u_1)$, each of which represents the number of vertices from s_{u_1} and s_{u_2} to their confluence point respectively.

We also introduce the term confluence group, which means a group of users traveling together.

Definition 2. If two users u_1, u_2 have a confluence point, they belong to the same **confluence group** on their common path from the confluence point to the destination. Given

a user u and an edge e in R_u, the confluence group is uniquely specified, and u travels on e together with the users in the confluence group. Let $C_u(i)$ be the confluence group that u belongs to in the i-th edge of R_u.

If two users u_1, u_2 have a confluence point, their paths from the confluence point to the destination must be identical. Therefore, we introduce the following constraints.

$$|R_{u_1}| - i(u_1, u_2) = |R_{u_2}| - i(u_2, u_1) \geq 0 \quad (1)$$

For $0 \leq k \leq |R_{u_1}| - i(u_1, u_2)$,

$$e_{u_1}(|R_{u_1}| - k) = e_{u_2}(|R_{u_2}| - k) \quad (2)$$

4. PROBLEM FORMULATION

In this paper, we do not simply consider lengths of edges on road networks; rather, we consider the traveling costs, which are the lengths of the edges multiplied by the weights, determined by the combination of users traveling together on the edges.

Definition 3. We define the traveling cost of user u traveling through edge e in a confluence group C as the length of e multiplied by a function $\alpha : (U, 2^U) \to \mathbb{R}^+$, which is called **weight function**.

Based on the above definitions, our problem is formulated as follows.

Input a set of users U, an initial position of each user s_u, and a destination v_T

Output a set of routes of each user u from s_u to v_T and a set of confluence points

Objective function (minimized)

$$\sum_{u \in U} \sum_{i=1}^{|R_u|} \alpha(u, C_u(i)) * l_{e_u(i)} \quad (3)$$

Output satisfies Equation (1), (2).

5. PROPOSED METHOD

In this proposed method, we consider a state in which a confluence group C exists at a vertex v. Let $dp(C, v)$ be the minimum total cost of each user u in C traveling from s_u to v based on the objective function defined as Equation (3). Let $R_u(v)$ be a route of a user u from s_u to a vertex v, and the exact definition of $dp(C, v)$ is as follows.

$$dp(C, v) = \min(\sum_{u \in C} \sum_{i=1}^{|R_u(v)|} \alpha(u, C_u(i)) * l_{e_u(i)}) \quad (4)$$

This proposed method is based on dynamic programming, which calculates $dp(C, v)$ for every confluence group C and vertex v. First, we consider the initial states in which each user u is located at s_u and belongs to the confluence group that consists of only u. Then, several transitions are performed for these states, and eventually, $dp(U, v_T)$ is calculated as the minimum value of the objective function.

The state in which a confluence group C exists at a vertex v arises through one of the two transitions below.

Confluence transition Two confluence groups C_1, C_2 merge together at a vertex v and turn into a single confluence group C.

Algorithm 1 Proposed Algorithm

```
1: for all C ⊂ U do
2:     for all v ∈ V do
3:         for all C_1 ∈ 2^C s.t. C_1 ≠ 0 and C_1 ≠ C do
4:             C_2 ← C \ C_1
5:             c ← dp(C_1, v) + dp(C_2, v)
6:             dp'(C, v) ← min(dp'(C, v), c)
7:     Q ← an empty priority queue
8:     for all v ∈ V do
9:         Q ← insert (dp'(C, v), v)
10:    while the size of Q > 0 do
11:        (c, v) ← a pair popped from Q
12:        if c < dp(C, v) then
13:            dp(C, v) ← c
14:            for all w ∈ V s.t. w is adjacent to v do
15:                c' ← c + traveling cost from v to w
16:                Q ← insert (c', w)
```

Traveling transition The confluence group C, which is formed at a vertex other than v through a confluence transition, travels to a vertex v through the shortest path.

Let $dp'(C, v)$ be the minimum cost of the state in which a confluence group C exists at a vertex v and which arises particularly through confluence transitions. Note that $dp(C, v)$ and $dp'(C, v)$ do not depend on the users who do not belong to C because we assume that the users in the same confluence group do not diverge. $dp(C, v)$ and $dp'(C, v)$ can be calculated as follows.

$$dp(C, v) = \min(dp'(C, w) + d(w, v) \mid w \in V) \quad (5)$$

Let $\{C_1, C_2\}$ be any partition of C,

$$dp'(C, v) = \min(dp(C_1, v) + dp(C_2, v)) \quad (6)$$

Equation (5) calculates traveling transitions, and given the values of $dp'(C, v)$ for all C and v, $dp(C, v)$ can be calculated using this formula. The specific steps of the calculation is explained in Algorithm 1. Equation (6) calculates confluence transitions, and given the values of $dp(C', v)$ for all v and C', which is a subset of C, $dp'(C, v)$ can be calculated using this formula. The optimum routes and confluence points can be obtained as the output of the problem by storing the transition in each calculation.

The pseudocode of the proposed method is shown in Algorithm 1. In this algorithm, a priority queue Q is introduced so that the order of the calculation of traveling transition can be dynamically changed. This enables our algorithm to be executed efficiently and applied to various applications. The worst-case time complexity of Algorithm 1 is $O(3^{|U|}|V| + 2^{|U|}|E| \log |V|)$.

6. EXPERIMENTS

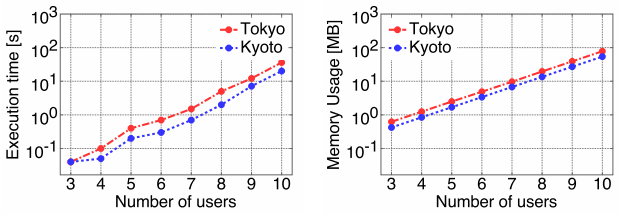
This section presents an evaluation of the performance of our method in terms of execution time and memory usage.

6.1 Data and Settings

We conduct our experiments using C++ (gcc 5.2.0) on a computer running Mac OSX 10.11.5 on an Intel Core i5

City	Vertices	Edges	Acquisition Year/Month
Tokyo	8901	20321	2016/03
Kyoto	6617	16658	2016/03

Table 1: Dataset



(a) Average Execution time (b) Memory usage

Figure 2: Performance of our algorithm

1600MHz CPU with 4GB of RAM. We export two road networks of the major cities in Japan, Tokyo and Kyoto from Open Street Map Japan[1]. We precompute the data obtained from Open Street Map Japan by excluding disconnected vertices. The basic information of the graphs after the precomputation is summarized in Table 1.

The weight function can return arbitrary positive real numbers; however, in this section, in order to reflect the real situations that we consider, we limit the values within the following range: $1/|C| \leq \alpha(u, C) < 1$. We choose the values randomly within the range for each query. Note that we set the values as we explained above simply to reflect real-world situations as faithfully as possible. Even if we set the values to any other, the execution time and memory usage of our algorithms do not change substantially. In this experiment, we generate 100 queries for each number of users and investigate the execution time and memory usage of our algorithms. For each query, the initial point of each user and the destination are generated randomly.

6.2 Experimental Results

Figure 2(a) shows that our algorithm is executable within ten seconds for less than 10 people on average. Since the number of users in the applications of our problem is relatively small, our method is sufficiently efficient for the application that we assume.

Figure 2(b) shows that our algorithm only requires less than a hundred MB of memory, which is sufficiently within the range of standard computers today.

7. DISCUSSIONS

Although our proposed exact method is intended for a relatively small number of users, approximation methods for a large number of users would be also useful.

The approximation methods for the MSTP[1][2] are difficult to apply to our problem. However, some approximation methods can be extended for the BaBP. In fact, we actually found that the approximation method proposed by Awerbuch and Azar[4] can be extended to our formulation with an additional constraint and this method guarantees an $O(\log^2 n)$ randomized approximation ratio. The details about this method are omitted due to space limitations.

[1]https://openstreetmap.jp/

The fact that we obtain the approximation method above gives an evidence that our problem is deeply related to the BaBP. We have already explained the relation between our problem and the MSTP: our formulation generalizes the MSTP although the time complexity of our exact method is the same as Dreyfus-Wagner Algorithm. In other words, we found interesting facts about the relation between the MSTP and the BaBP, although these two problems have not been associated with each other to the best of our knowledge.

8. CONCLUSION

This paper highly generalized the previous works on multi-user routing optimization with confluence and proposed an efficient exact method for the problem. A large-scale experiment demonstrated that our proposed method is practically efficient. For future work, we plan to propose formulation with some constraints related to time. Moreover, we also plan to propose practical and theoretical approximation methods to construct a reliable system for a large number of users.

9. REFERENCES

[1] Winter P, “Steiner problem in networks”, a survey. Networks 17(2), pp. 129–167, 1987.

[2] Prömel, Hans Jürgen, Steger, Angelika, “The Steiner tree problem: a tour through graphs, algorithms, andcomplexity”, Advanced Lectures in Mathematics, Basics 3: Complexity, pp. 41-62, 2002.

[3] S. E. Dreyfus, R. A. Wagner, “The Steiner problem in graphs”, Networks, 1, pp. 195–207, 1972.

[4] Awerbuch B, Azar Y, “Buy-at-Bulk Network Design”, Proceedings of IEEE FOCS, pp. 542–547, 1997.

[5] Ashish G, Ian P, “An Oblivious O(1)-Approximation for Single Source Buy-at-Bulk”, Proceedings of IEEE FOCS, pp. 442–450, 2009.

[6] Srinivasagopalan S, Busch C, Iyengar, “An Oblivious Spanning Tree for Single-Sink Buy-at-Bulk in Low Doubling-Dimension Graphs”, IEEE Transactions on Computers, Volume: 61, Issue: 5, pp. 700–712, 2012.

[7] Bjoern Z, Alexander M, “Calculating Meeting Points for Multi User Pedestrian Navigation Systems”, Proceedings of Advances in Artificial Intelligence, pp. 347–356, 2011.

[8] Zhang X, Asano Y, Yoshikawa M, “Mutually beneficial Confluent Routing”, IEEE Transactions on Knowledge and Data Engineering, Volume: 28, No. 10, pp. 2681-2696, 2016.

[9] Savelsbergh M. W. P, Sol M, “The general pickup and delivery problem”, Transportation Science, Volume: 29, pp. 17–29, 1995.

[10] Cao B, Alarabi L, Mokbel M. F, Basalamah A, “A scalable dynamic ride sharing system”, Proceedings of 16th MDM, pp. 4–13, 2015.

[11] Drews F, Luxen D, “Multi-hop ride sharing”, Proceedings of 6th SoCS, pp. 71–79, 2013.

[12] Geisberger R, Luxen D, Neubauer S, Sanders P, Volker L, “Fast detour computation for ride sharing”, 10th Workshop on Algorithmic Approaches for Transportation Modelling Optimization and Systems, pp 88–99, 2009.