Scientific Research Publishing

Journal of Applied Mathematics and Physics, 2017, 5, 1678-1685
http://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

A Dynamic Programming Algorithm for the Ridersharing Problem Restricted with Unique Destination and Zero Detour on Trees

Yiming Li[1], Huiqiang Lu[2], Zhiqian Ye[3], Xiao Zhou[4]

[1]Wenzhou University, Wenzhou, China
[2]Zhejiang University of Technology, Hangzhou, China
[3]Zhejiang University, Hangzhou, China
[4]Tohoku University, Sendai, Japan
Email: ymli@wzu.edu.cn, lhq@zjut.edu.cn, yezhiqian@zju.edu.cn, zhou@ecei.tohoku.ac.jp

How to cite this paper: Li, Y.M., Lu, H.Q., Ye, Z.Q. and Zhou, X. (2017) A Dynamic Programming Algorithm for the Ridersharing Problem Restricted with Unique Destination and Zero Detour on Trees. *Journal of Applied Mathematics and Physics*, **5**, 1678-1685.
https://doi.org/10.4236/jamp.2017.59140

Received: August 7, 2017
Accepted: September 12, 2017
Published: September 15, 2017

Abstract

We deal with the problem of sharing vehicles by individuals with similar itineraries which is to find the minimum number of drivers, each of which has a vehicle capacity and a detour to realize all trips. Recently, Gu *et al.* showed that the problem is NP-hard even for star graphs restricted with unique destination, and gave a polynomial-time algorithm to solve the problem for paths restricted with unique destination and zero detour. In this paper we will give a dynamic programming algorithm to solve the problem in polynomial time for trees restricted with unique destination and zero detour. In our best knowledge it is a first polynomial-time algorithm for trees.

Keywords

Dynamic Programming Algorithm, Rideshare, Tree

1. Introduction

There are many previous researches for assigning passengers to drivers [1]-[7]. In this paper we deal with a more general ridesharing problem, introduced by Gu *et al.* [8], in which each individual is a driver or passenger. Let G be a connected and edge-weighted graph with vertex set $V(G)$ and edge set $E(G)$. Let k be a positive integer, and let a *ridesharing infomation* R be a set of k trips $(i, s_i, t_i, c_i, d_i, P_i)$, $1 \le i \le k$, where

- s_i and t_i are the source (start location) and the destination, respectively, of the i-th trip;

- c_i is the number of seats (capacity) of the i-th driver available for passengers including the driver;
- d_i is the detour distance limit which the i-th driver can tolerate for offering ridesharing services; and
- P_i is the preferred path of the i-th trip from s_i to t_i in G.

Let $I(R)$ be the set of the indices of all trips in R which should be served, that is, $I(R)=\{i \mid (i,s_i,t_i,c_i,d_i,P_i)\in R\}$. A *ridesharing of a graph G with R* is a mapping $f: I(R)\to I(R)$ such that, for each $i\in I(R)$, if $f^{-1}(i)\neq\varnothing$ then $i\in f^{-1}(i)$, $|f^{-1}(i)|\leq c_i$ and there is a trip (walk) P satisfying $P_j\subseteq P$ for each $j\in f^{-1}(i)$ and the distance of P is at most d_i plus the distance of P_i, where $f^{-1}(i)=\{j \mid f(j)=i\}$, that is, $f^{-1}(i)$ is the set of the indices of trips served by the i-th driver. Let $\mathbf{driver}(f)$ be the number of drivers of a ridesharing f, that is,

$$\mathbf{driver}(f)=\left|\left\{i \mid f^{-1}(i)\neq\varnothing\right\}\right|.$$

A ridesharing f of G with R is *optimal* if $\mathbf{driver}(f)$ is minimum among all ridesharings f. The ridesharing problem is to find an optimal ridesharing f for a given graph G and a ridesharing information R.

Recently, Gu *et al.* showed that the problem is NP-hard even for star graphs restricted with unique destination, and gave a polynomial-time algorithm to solve the problem for paths restricted with unique destination and zero detour [8]. In this paper we will give a dynamic programming algorithm to solve the problem for trees T restricted with unique destination and zero detour. Our algorithm runs in time $O(k^2n)$, where k is the number of the trips and n is the number of the vertices in T. In our best knowledge it is a first polynomial-time algorithm for trees.

2. A Dynamic Programming Algorithm

The ridesharing problem is NP-hard even for star graphs restricted with unique destination if some detours are not zero [8]. However, in this section, we have the following theorem for the problem of trees restricted with unique destination and zero detour.

The ridesharing problem of trees T can be solved in time $O(k^2n)$ if the destinations of all the k trips are same and the detour of each driver is zero, where n is the number of the vertices in T.

In the remainder of this section we give an algorithm to solve the ridesharing problem for trees in polynomial time as a proof of Theorem 2. Let k be a positive integer and let R be a set of the k trips (i,s_i,t_i,c_i,d_i,P_i), $1\leq i\leq k$, as an instance of the ridesharing problem of T. Since we deal with the problem restricted with unique destination and zero detours, let r be the unique destination. Then for each i, $1\leq i\leq k$, we have $t_i=r$ and $d_i=0$, and P_i is the unique path in T. For the sake of notational convenience we redefine the ridesharing information $R=\{(i,s_i,c_i) \mid 1\leq i\leq k\}$, and let (T,R,r) be an instance of the ridesharing

problem of T with the ridesharing information R and the unique destination r in the remainder of this paper.

Since the destinations of all trips are same, we choose the destination r as the root of a given tree T, and regard T as rooted tree. For each vertex v of T, we denote by T_v the subtree of T which is rooted at v and is induced by all descendants of v in T. For each edge $e=(v,v')\in E(T)$ such that v is the parent of v' in T, we denote by T_e the subtree of T which is rooted at v and is induced by v, v' and the descendants of v' in T. For a subtree T' of T rooted at v, let

$$R(T',v)=\{(i,s_i,c_i)\in R \mid s_i\in V(T')\setminus\{v\}\}.$$

For each α, $0\le\alpha\le k$, we denote by $\mathbf{cap}(T',v,\alpha)$ the maximum number of additional trips which can be served after serving all trips in $R(T',v)$ using at most α drivers, that is,

$$\mathbf{cap}(T',v,\alpha)=\max_f \mathbf{cap}(f)$$

where the maximum is taken over all ridesharings f with the instance $(T',R(T',v),v)$ satisfying $\mathbf{driver}(f)\le\alpha$ and

$$\mathbf{cap}(f)=\sum_{i\in I(R(T',v)),\, f^{-1}(i)\neq\varnothing} c_i-\left|R(T',v)\right|.$$

Let $\mathbf{cap}(T',v,\alpha)=-\infty$ if α drivers can not serve all trips in $R(T',v)$ with the unique destination v. The main step of our algorithm is to compute the table of the functions $\mathbf{cap}$ from leaves to the root r of T by dynamic programming. From the table on the root r, it is obvious that the minimum α satisfying $\mathbf{cap}(T_r,r,\alpha)\neq-\infty$ is the minimum number of drivers to serve all the trips. Although we give a dynamic programming algorithm to compute the minimum number of drivers to serve all the trips, the algorithm can be easily modified to actually find an optimal ridesharing f. We thus show how to compute such all the tables on vertices v from leaves to the root in the remainder of this section.

2.1. The Vertex *v* Is a Leaf in *T*

In this case, since v is a leaf in T, there is no trips from v's descendants in T_v. Therefore, by the definition of $\mathbf{cap}(T_v,v,\alpha)$, we trivially have the following lemma.

Let v be an arbitrary leaf of T. Then $\mathbf{cap}(T_v,v,\alpha)=0$ for each $\alpha\ge0$.

2.2. The Vertex *v* Is an Internal Vertex in *T*

In this case v is an internal vertex in T. Let $v_1,v_2,\cdots,v_l$ be the children of v ordered arbitrarily, and let e_j, $1\le j\le l$, be the edge joining v to v_j, as illustrated in **Figure 1**. The subtree T_{v_j}, $1\le j\le l$, of T is rooted at v_j and is induced by all descendants of v_j in T. We denote by T_v^j the subtree of T which consists of the vertex v, the edges $e_1,e_2,\cdots,e_j$ and the subtrees $T_{v_1},T_{v_2},\cdots,T_{v_j}$. T_v^{j-1} is indicated by a dotted line in **Figure 1**. Obviously $T_v=T_v^l$.

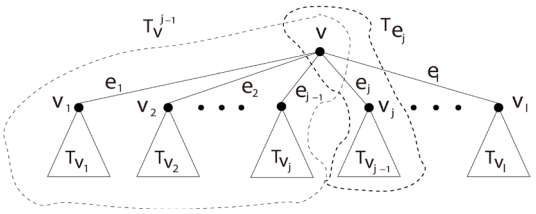

Figure 1. Subtrees T_v, T_v^{j-1} and T_{e_j} rooted at *v*.

We first compute the function $\mathbf{cap}(T_{e_j}, v, \alpha)$ for all α and j, $0 \le \alpha \le k$ and $1 \le j \le l$, as in the following lemma 1, where T_{e_j} is the subtree obtained from T_{v_j} by adding the edge e_j, indicated by a dotted thick line in **Figure 1**.

Let $R(v_j) = \{(i, s_i, c_i) \in R \mid s_i = v_j\}$ be the subset of *R* such that the start location is at v_j, and let

$$\gamma(T_{e_j}, v, \alpha) = \max_{0 \le \alpha_1 \le \alpha} \left\{ \mathbf{cap}(T_{v_j}, v_j, \alpha_1) + \max_{I' \subseteq I(R(v_j)), |I'| = \alpha - \alpha_1} \sum_{i \in I'} c_i - |R(v_j)| \right\}. \quad (1)$$

Then

$$\mathbf{cap}(T_{e_j}, v, \alpha) = \begin{cases} \gamma(T_{e_j}, v, \alpha) & \text{if } \gamma(T_{e_j}, v, \alpha) \ge 0; \\ -\infty & \text{otherwise.} \end{cases} \quad (2)$$

Furthermore, all $\mathbf{cap}(T_{e_j}, v, \alpha)$, $0 \le \alpha \le k$, can be computed from all $\mathbf{cap}(T_{v_j}, v_j, \alpha_1)$, $0 \le \alpha_1 \le k$, in time $O(k^2)$.

Proof. We first prove

$$\mathbf{cap}(T_{e_j}, v, \alpha) \le \gamma(T_{e_j}, v, \alpha). \quad (3)$$

If $\mathbf{cap}(T_{e_j}, v, \alpha) = -\infty$, then Equation (3) trivially holds. We thus assume that $\mathbf{cap}(T_{e_j}, v, \alpha) \ne -\infty$, that is, there is a ridesharing *f* with the instance $(T_{e_j}, R(T_{e_j}, v), v)$ such that $\mathbf{driver}(f) \le \alpha$ and $\mathbf{cap}(T_{e_j}, v, \alpha) = \mathbf{cap}(f)$. Let f' be a mapping from $I(R(T_{v_j}, v_j))$ to $I(R(T_{v_j}, v_j))$ such that $f'(i) = f(i)$ for each $i \in I(R(T_{v_j}, v_j))$. Then clearly f' is a ridesharing of T_{v_j} with the instance $(T_{v_j}, R(T_{v_j}, v_j), v_j)$ since *f* is a ridesharing of T_{e_j} and T_{v_j} is a subtree of T_{e_j}. Let $\alpha_1 = \mathbf{driver}(f')$, then by the definition of $\mathbf{cap}(T_{v_j}, v_j, \alpha_1)$, we have $\mathbf{cap}(f') \le \mathbf{cap}(T_{v_j}, v_j, \alpha_1)$. Let

$$I' = \{i \in I(R(v_j)) \mid f^{-1}(i) \ne \varnothing\}.$$

Then the $\mathbf{driver}(f)$ drivers in *f* consists of the $\mathbf{driver}(f')$ drivers from start locations in T_{v_j} and the $|I'|$ drivers from start locations at v_j, and hence $\mathbf{driver}(f) = \mathbf{driver}(f') + |I'|$. Therefore, we have $\alpha = \alpha_1 + |I'|$ and

$$\begin{aligned} \mathbf{cap}(T_{e_j}, v, \alpha) = \mathbf{cap}(f) &= \sum_{i \in I(R(T_{e_j}, v)), f^{-1}(i) \ne \varnothing} c_i - \left| R(T_{e_j}, v) \right| \\ &= \sum_{i \in I(R(T_{v_j}, v_j)), f^{-1}(i) \ne \varnothing} c_i + \sum_{i \in I(R(v_j)), f^{-1}(i) \ne \varnothing} c_i - \left| R(T_{v_j}, v_j) \right| - \left| R(v_j) \right| \\ &= \mathbf{cap}(f') + \sum_{i \in I'} c_i - |R(v_j)| \le \mathbf{cap}(T_{v_j}, v_j, \alpha_1) + \sum_{i \in I'} c_i - |R(v_j)| \\ &\le \gamma(T_{e_j}, v, \alpha), \end{aligned}$$

verified Equation (3).

We next prove that

$$\mathbf{cap}(T_{e_j}, v, \alpha) \geq \gamma(T_{e_j}, v, \alpha) \tag{4}$$

if $\gamma(T_{e_j}, v, \alpha) \geq 0$. By Equation (1) m choose α_1 and I' such that $0 \leq \alpha_1 \leq \alpha$, $I' \subseteq I(R(v_j))$, $|I'| = \alpha - \alpha_1$ and

$$\gamma(T_{e_j}, v, \alpha) = \mathbf{cap}(T_{v_j}, v_j, \alpha_1) + \sum_{i \in I'} c_i - |R(v_j)|. \tag{5}$$

Since $\gamma(T_{e_j}, v, \alpha) \geq 0$, we have $\mathbf{cap}(T_{v_j}, v_j, \alpha_1) \neq -\infty$, and hence there is a ridesharing f' with the instance $(T_{v_j}, R(T_{v_j}, v_j), v_j)$ such that $\mathbf{driver}(f') \leq \alpha_1$ and $\mathbf{cap}(f') = \mathbf{cap}(T_{v_j}, v_j, \alpha_1)$. One can obtain a ridesharing f with the instance $(T_{e_j}, R(T_{e_j}, v), v)$ from f' using α_1 drivers in f' plus the new $|I'|$ drivers having the trip indices in I'. By Equation (5) we have $\mathbf{cap}(f) = \gamma(T_{e_j}, v, \alpha)$, and by the definition $\mathbf{cap}(T_{e_j}, v, \alpha) \geq \mathbf{cap}(f)$, and hence we have $\mathbf{cap}(T_{e_j}, v, \alpha) \geq \gamma(T_{e_j}, v, \alpha)$, verified Equation (4).

By Equations (3) and (4) m Equation (2) holds true. We finally show that for each α, $0 \leq \alpha \leq k$, $\mathbf{cap}(T_{e_j}, v, \alpha)$, can be computed from all $\mathbf{cap}(T_{v_j}, v_j, \alpha_1)$, $0 \leq \alpha_1 \leq k$, in time $O(k^2)$ as follows.

By Equation (1) for a given α_1, clearly I' is the set of the $\alpha - \alpha_1$ indices $i \in I(R(v_j))$ such that c_i is at least $\alpha - \alpha_1$ largest among all $c_{i'}$, $i' \in I(R(v_j))$. One can sorted all c_i, $i \in I(R(v_j))$, in non-increasing order, and compute all prefix sum of c_i. This can be done in time $O(k \log k)$. Then, for any given pair of α and α_1, $0 \leq \alpha_1 \leq \alpha \leq k$,

$$\max_{I' \subseteq I(R(v_j)), |I'| = \alpha - \alpha_1} \sum_{i \in I'} c_i$$

can be computed in $O(1)$ time. One thus can compute $\gamma(T_{e_j}, v, \alpha)$ in time $O(k)$ for each α since $\alpha_1 \leq \alpha \leq k$, and hence $\mathbf{cap}(T_{e_j}, v, \alpha)$ for all α, $0 \leq \alpha \leq k$, can be computed from all $\mathbf{cap}(T_{v_j}, v_j, \alpha_1)$ in time $O(k^2)$. □

We finally compute the function $\mathbf{cap}(T_v^j, v, \alpha)$ for all α and j, $0 \leq \alpha \leq k$ and $1 \leq j \leq l$, as in the following lemma.

For each α, $0 \leq \alpha \leq k$,

$$\mathbf{cap}(T_v^j, v, \alpha) = \max_{0 \leq \alpha_1, \alpha_2 \leq \alpha,\ \alpha_1 + \alpha_2 = \alpha} \left\{ \mathbf{cap}(T_v^{j-1}, v, \alpha_1) + \mathbf{cap}(T_{e_j}, v, \alpha_2) \right\}.$$

Furthermore, all $\mathbf{cap}(T_v^j, v, \alpha)$, $0 \leq \alpha \leq k$, can be computed from all $\mathbf{cap}(T_v^{j-1}, v, \alpha_1)$ and $\mathbf{cap}(T_{v_j}, v_j, \alpha_2)$ in time $O(k^2)$.

Proof. We first prove

$$\mathbf{cap}(T_v^j, v, \alpha) \leq \max_{0 \leq \alpha_1, \alpha_2 \leq \alpha,\ \alpha_1 + \alpha_2 = \alpha} \left\{ \mathbf{cap}(T_v^{j-1}, v, \alpha_1) + \mathbf{cap}(T_{e_j}, v, \alpha_2) \right\}. \tag{6}$$

If $\mathbf{cap}(T_v^j, v, \alpha) = -\infty$, then Equation (6) trivially holds. We thus assume that $\mathbf{cap}(T_v^j, v, \alpha) \neq -\infty$, that is, there is a ridesharing f with the instance $(T_v^j, R(T_v^j, v), v)$ such that $\mathbf{driver}(f) \leq \alpha$ and $\mathbf{cap}(T_v^j, v, \alpha) = \mathbf{cap}(f)$. Let f_1 be a mapping from $I(R(T_v^{j-1}, v))$ to $I(R(T_v^{j-1}, v))$ such that $f_1(i) = f(i)$ for each $i \in I(R(T_v^{j-1}, v))$. Then clearly f_1 is a ridesharing of T_v^{j-1} with the

instance $(T_v^{j-1}, R(T_v^{j-1}, v), v)$. Let $\alpha_1 = \mathbf{driver}(f_1)$. By the definition of $\mathbf{cap}(T_v^{j-1}, v, \alpha_1)$ we have $\mathbf{cap}(f_1) \le \mathbf{cap}(T_v^{j-1}, v, \alpha_1)$. Similarly, let f_2 be a mapping from $I(R(T_{e_j}, v))$ to $I(R(T_{e_j}, v))$ such that $f_2(i) = f(i)$ for each $i \in I(R(T_{e_j}, v))$. Then clearly f_2 is a ridesharing of T_{e_j} with the instance $(T_{e_j}, R(T_{e_j}, v), v)$. Let $\alpha_2 = \mathbf{driver}(f_2)$. By the definition of $\mathbf{cap}(T_{e_j}, v, \alpha_2)$ we have $\mathbf{cap}(f_2) \le \mathbf{cap}(T_{e_j}, v, \alpha_2)$. Furthermore, $\mathbf{cap}(f) = \mathbf{cap}(f_1) + \mathbf{cap}(f_2)$ and $\mathbf{driver}(f) = \mathbf{driver}(f_1) + \mathbf{driver}(f_2)$, and hence $\alpha = \alpha_1 + \alpha_2$. We thus have

$$\begin{aligned}\mathbf{cap}(T_v^j, v, \alpha) &= \mathbf{cap}(f) = \mathbf{cap}(f_1) + \mathbf{cap}(f_2)\\ &\le \mathbf{cap}(T_v^{j-1}, v, \alpha_1) + \mathbf{cap}(T_{e_j}, v, \alpha_2),\end{aligned}$$

verified Equation (6).

We next prove

$$\mathbf{cap}(T_v^j, v, \alpha) \ge \max_{0 \le \alpha_1, \alpha_2 \le \alpha,\ \alpha_1 + \alpha_2 = \alpha} \left\{ \mathbf{cap}(T_v^{j-1}, v, \alpha_1) + \mathbf{cap}(T_{e_j}, v, \alpha_2) \right\}. \quad (7)$$

Choose α_1' and α_2' such that $0 \le \alpha_1', \alpha_2' \le \alpha$, $\alpha_1' + \alpha_2' = \alpha$ and $\mathbf{cap}(T_v^{j-1}, v, \alpha_{1'}) + \mathbf{cap}(T_{e_j}, v, \alpha_{2'}) = \max_{0 \le \alpha_1, \alpha_2 \le \alpha,\ \alpha_1 + \alpha_2 = \alpha} \left\{ \mathbf{cap}(T_v^{j-1}, v, \alpha_1) + \mathbf{cap}(T_{e_j}, v, \alpha_2) \right\}$.

We may assume $\mathbf{cap}(T_v^{j-1}, v, \alpha_1') \ne -\infty$ and $\mathbf{cap}(T_{e_j}, v, \alpha_2') \ne -\infty$; otherwise Equation (7) trivially holds.

Since $\mathbf{cap}(T_v^{j-1}, v, \alpha_1') \ne \infty$, there is a ridesharing f_1 with the instance $(T_v^{j-1}, R(T_v^{j-1}, v), v)$ such that $\mathbf{cap}(f_1) = \mathbf{cap}(T_v^{j-1}, v, \alpha_1')$. Similarly, since $\mathbf{cap}(T_{e_j}, v, \alpha_2') \ne -\infty$, there is a ridesharing f_2 with the instance $(T_{e_j}, R(T_{e_j}, v), v)$ such that $\mathbf{cap}(f_2) = \mathbf{cap}(T_{e_j}, v, \alpha_2')$. Let f be a mapping such that for each $i \in I(R(T_v^j, v))$

$$f(i) = \begin{cases} f_1(i) & \text{if } i \in I(R(T_v^{j-1}, v)),\\ f_2(i) & \text{otherwise, that is, } i \in I(R(T_{e_j}, v)).\end{cases}$$

Then clearly f is a ridesharing of T_v^j with the instance $(T_v^j, R(T_v^j, v), v)$ such that $\mathbf{cap}(f) = \mathbf{cap}(f_1) + \mathbf{cap}(f_2)$ and $\mathbf{driver}(f) = \mathbf{driver}(f_1) + \mathbf{driver}(f_2)$. By the definition of $\mathbf{cap}(T_v^j, v, \alpha)$, we have

$$\begin{aligned}\mathbf{cap}(T_v^j, v, \alpha) &\ge \mathbf{cap}(f) = \mathbf{cap}(f_1) + \mathbf{cap}(f_2)\\ &= \mathbf{cap}(T_v^{j-1}, v, \alpha_1') + \mathbf{cap}(T_{e_j}, v, \alpha_2')\\ &= \max_{0 \le \alpha_1, \alpha_2 \le \alpha,\ \alpha_1 + \alpha_2 = \alpha} \left\{ \mathbf{cap}(T_v^{j-1}, v, \alpha_1) + \mathbf{cap}(T_{e_j}, v, \alpha_2) \right\},\end{aligned}$$

verified Equation (7).

Furthermore, clearly $\mathbf{cap}(T_v^j, v, \alpha)$ for all α and j, $0 \le \alpha \le k$, can be computed in time $O(k^2)$ from all $\mathbf{cap}(T_v^{j-1}, v, \alpha_1)$ and $\mathbf{cap}(T_{e_j}, v, \alpha_2)$. □

2.3. Algorithm

From Lemmas 2.1—1 one can obtain the following algorithm to compute all $\mathbf{cap}(T_v, v, \alpha)$, $v \in V(T)$ and $0 \le \alpha \le k$.

Algorithm Alg(T_v, R, v, cap) begin if v is a leaf in T, then $cap(T_v, v, \alpha) = 0$ for all α, $0 \le \alpha \le k$, by Lemm 2.1; else if v is an internal vertex in T, then begin let $v_1, v_2, \cdots, v_l$ be the children of v ordered arbitrarily, and let e_j, $1 \le j \le l$, be the edge joining v to v_j; for each j, $1 \le j \le l$, compute all $cap(T_{e_j}, v, \alpha)$, $0 \le \alpha \le k$, from all $cap(T_{v_j}, v_j, \alpha_1)$ m $0 \le \alpha_1 \le k$, by Lemma 1; for each j, $1 \le j \le l$, compute all $cap(T_v^j, v, \alpha)$, $0 \le \alpha \le k$, from all $cap(T_v^{j-1}, v, \alpha_1)$, $0 \le \alpha_1 \le k$, and all $cap(T_{e_j}, v, \alpha_2)$, $0 \le \alpha_2 \le k$, by Lemma 1; let $cap(T_v, v, \alpha) = cap(T_v^l, v, \alpha)$ for all α, $0 \le \alpha \le k$; end end

Clearly it runs in time $O(k^2 n)$ since T has the n vertices. This completes to prove Theorem 2.

3. Conclusion

In this paper we gave a dynamic programming algorithm to solve the ridesharing problem for trees restricted with unique destination and zero detour. Our algorithm runs in polynomial time. However, it is still open whether or not there is a polynomial-time algorithm to solve the problem restricted with unique destination and zero detour for series-parallel graphs and graphs with bounded treewidth.

Funding

This work is partially supported by JSPS KAKENHI Grant Number JP16K00003 (X. Zhou).

References

[1] Agatz, N., Erera, A., Savelsbergh, M. and Wang, X. (2011) Dynamic Ride-Sharing: A Simulation Study in Metro Atlanta. *Transp. Res. Part B*, **45**, 1540-1564. https://doi.org/10.1016/j.trb.2011.05.017

[2] Agatz, N., Erera, A., Savelsbergh, M. and Wang, X. (2012) Optimization for Dynamic Ride-Sharing: A Review. *European Journal of Operational Research*, **223**, 295-303. https://doi.org/10.1016/j.ejor.2012.05.028

[3] Baldacci, R., Maniezzo, V. and Mingozzi, A. (2004) An Exact Method for the Car Pooling Problem Based on Lagrangean Column Generation. *Oper. Res.*, **52**, 422-439. https://doi.org/10.1287/opre.1030.0106

[4] Brucker, P. and Nordmann, L. (1994) The *k*-Track Assignment Problem. *Computing*, **54**, 97-122. https://doi.org/10.1007/BF02238071

[5] Chan, N.D. and Shaheen, S.A. (2012) Ridesharing in North America: Past, Present, and Future. *Transp. Rev.*, **32**, 93-112. https://doi.org/10.1080/01441647.2011.621557

[6] Cordeau, J.-F. and Laporte, G. (2007) The Dial-a-Ride Problem: Models and Algorithms. *Annals of Operations Research*, **153**, 29-46. https://doi.org/10.1007/s10479-007-0170-8

[7] Ghoseiri, K., Haghani, A. and Hamedi, M. (1979) Real-Time Ridesharing Matching Problem. Final Report of UMD-2009-05. U.S. Department of Transportation.

[8] Gu, Q.-P., Liang, L. and Zhang, G. (2016) Algorithmic Analysis for Ridesharing of

Personal Vehicles. *Proc. of COCOA* 2016, LNCS 10043, 438-452.
https://doi.org/10.1007/978-3-319-48749-6_32