Journal of Combinatorial Optimization (2023) 45:104
https://doi.org/10.1007/s10878-023-01029-2

An approximation algorithm for the clustered path travelling salesman problem

Jiaxuan Zhang[1] · Suogang Gao[1] · Bo Hou[1] · Wen Liu[1]

Accepted: 3 April 2023 / Published online: 30 April 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
In this paper, we consider the clustered path travelling salesman problem. In this problem, we are given a complete graph $G = (V, E)$ with an edge weight function w satisfying the triangle inequality. In addition, the vertex set V is partitioned into clusters $V_1, \ldots, V_k$ and s, t are two given vertices of G with $s \in V_1$ and $t \in V_k$. The objective of the problem is to find a minimum Hamiltonian path of G from s to t, where all vertices of each cluster are visited consecutively. In this paper, we deal with the case that the start-vertex and the end-vertex of the path on each cluster are both specified, and for it we provide a polynomial-time approximation algorithm.

Keywords Travelling salesman problem · Stacker crane problem · Path · Cluster

1 Introduction

The travelling salesman problem (TSP) is a best-known combinatorial optimization problem. In this problem, we are given a complete graph $G = (V, E)$ with an edge weight function satisfying the triangle inequality. The task of the TSP is to find a minimum Hamiltonian cycle. This problem is NP-hard and has multitudes of applications (Bland and Shallcross 1989; Christofides 1976).

Meanwhile, TSP has quite a lot variants (Grötschel and Holland 1991; Plante et al. 1987). Among these variants there is an important one called the path travelling salesman problem (PTSP), whose task is to find a minimum Hamiltonian path. For the PTSP, Hoogeveen (1991) presented a $\frac{5}{3}$-approximation algorithm. An et al. (2015) improved this result and gave a $\frac{1+\sqrt{5}}{2}$-approximation algorithm. Recently, Zenklusen (2019) developed the best known $\frac{3}{2}$-approximation algorithm. For more work on this

✉ Wen Liu
liuwen1975@126.com

[1] School of Mathematical Sciences, Hebei Key Laboratory of Computational Mathematics and Applications, Hebei Normal University, Shijiazhuang 050024, People's Republic of China

problem, one can see (Gottschalk and Vygen 2018; Sebő and van Zuylen 2016; Traub and Vygen 2019).

In the TSP, the delivery locations are supposed to have the same degree of urgencies, that is, all vertices can be visited in any order. However, in a number of real-world routing applications, different levels of priorities at the delivery locations need to be taken into account (Jacobson et al. 2012; Yang and Feng 2013; Zhu et al. 2019). Locations requiring the same level of urgency can be put in a common cluster and different clusters have distinct priorities. Such a problem is called the ordered cluster travelling salesman problem. Anily et al. (1999) designed an approximation algorithm for the ordered cluster travelling salesman problem with a performance guarantee of $\frac{5}{3}$. Since visiting in the specified order can lead to an inefficient route in terms of traveling cost, a more relaxed version is considered where delivery locations within the same cluster must be visited consecutively but there is no priority associated to a cluster. This leads to the fact that we can visit clusters in any order and this variant of TSP is called the clustered travelling salesman problem (CTSP). To be formal, in the CTSP, we are given an undirected complete graph $G = (V, E)$ with the vertex set V being partitioned into clusters $V_1, \cdots, V_k$, and the goal is to find a minimum Hamiltonian cycle of G, where all vertices of each cluster are visited consecutively. Note that if $k = 1$, the CTSP is exactly the TSP. So in the following we assume $k \geq 2$. Chisman (1975) first introduced the CTSP and gave some applications about it. Arkin et al. (1994) developed the first approximation algorithm for the CTSP with a performance guarantee of $\frac{7}{2}$. Guttmann-Beck et al. (2000) designed approximation algorithms for several cases of the CTSP by decomposing them into the PTSP together with the stacker crane problem, or the PTSP together with the rural postman problem. Then, Kawasaki and Takazawa (2020) improved the approximation ratios by applying the approximation algorithm for the PTSP given by Zenklusen (2019). Applications and other related work for the CTSP can be found in Gendreau et al. (1996); Jongens and Volgenant (1985).

Motivated by the work of Guttmann-Beck et al. (2000), Frederickson et al. (1978) and Kawasaki and Takazawa (2020), we study the clustered path travelling salesman problem (CPTSP). In the CPTSP, we are given a complete graph $G = (V, E)$ with an edge weight function w satisfying the triangle inequality. The vertex set V is partitioned into clusters $V_1, \cdots, V_k$ and s, t are two given vertices of G with $s \in V_1$ and $t \in V_k$. The goal is to find a minimum Hamiltonian path of G from s to t, where all vertices of each cluster are visited consecutively. Obviously, s is the start-vertex of the Hamiltonian path and t is the end-vertex of the Hamiltonian path. We assume $s \in V_1$ and $t \in V_k$, so V_1 and V_k are the first and the last clusters to be visited respectively. For other $k-2$ clusters, we visit them in any order. The case that the visiting order of these $k-2$ clusters is also given was studied by Anily et al. (1999). Note that a Hamiltonian path of the CPTSP induces a Hamiltonian path on each cluster. For simplicity, we assume that the start-vertex and the end-vertex of the induced Hamiltonian path on each cluster are both specified. In this paper, we design an approximation algorithm with an approximation ratio $\frac{8}{3}$ for the CPTSP by decomposing it into the path travelling salesman problem and the path version of the stacker crane problem.

The paper is organized as follows. In Sect. 2, we give some definitions and results. In Sect. 3, by modifying the algorithms for the stacker crane problem given in Fred-

erickson et al. (1978), we design two algorithms for the path stacker crane problem. Based on these two algorithms, we design an algorithm for the clustered path travelling salesman problem and analyze its approximation ratio in Sect. 4. In Sect. 5, we provide several future research problems.

2 Preliminaries

In this section, we introduce some terminology, concepts and related results.

A graph G is a pair (V, E) where V is a set of objects called vertices and E is a set of edges. Each edge is a pair $\{u, v\}$ of vertices $u, v \in V$. We call $\{u, v\}$ the undirected edge. The degree of a vertex is the number of edges incident on the vertex. A vertex is of even degree if the degree is an even number, and is of odd degree otherwise. If an undirected edge $\{u, v\}$ is associated with a direction, a directed edge (arc) is obtained and we denote it by (u, v) if the direction is from u to v and (v, u) otherwise. Vertices u and v are called the tail and the head of the arc (u, v), respectively. A directed graph has a set of vertices and a set of directed edges. In a directed graph, the out-degree (in-degree) of a vertex is the number of directed edges directed out of (into) the vertex. A mixed graph is a triple (V, E, D) where V is a set of vertices, E is a set of undirected edges and D is a set of directed edges. A mixed multigraph is a graph, possibly with parallel edges, each of which is either undirected or directed. In a mixed (multi)graph, the degree of a vertex is the number of edges and arcs incident on the vertex.

A walk in a graph G from vertex v_1 to v_l is a sequence $(v_1, \{v_1, v_2\}, v_2, \{v_2, v_3\}, v_3, \cdots, v_{l-1}, \{v_{l-1}, v_l\}, v_l)$, in which all $v_i's$ are vertices and $\{v_{i-1}, v_i\} \in E$ for $i = 2, 3, \cdots, l$. We call v_1, v_l the start-vertex and the end-vertex of the walk. A path is a walk with no repeated vertices, a trail is a walk with no repeated edges, and an Eulerian trail is a walk passing through every edge exactly once. If an Eulerian trail is closed (the start-vertex is the same as the end-vertex), it is called an Eulerian tour. A Hamiltonian path/cycle of G is a path/cycle visiting every vertex of G exactly once. A directed walk in a directed graph is a sequence $(v_1, (v_1, v_2), v_2, (v_2, v_3), v_3, \cdots, v_{l-1}, (v_{l-1}, v_l), v_l)$, a directed path is a directed walk with no repeated vertices, a directed trail is a directed walk with no repeated arcs, and a directed Eulerian trail is a directed walk passing through every arc exactly once.

In the stacker crane problem (SCP), we are given a mixed multigraph $G' = (V', E', D)$, where $V' = \{s_i, t_i \mid i \in [k]\}$, (V', E') is an undirected complete graph with an edge weight function w satisfying the triangle inequality, and $D = \{(s_i, t_i) \mid i \in [k]\}$. The objective is to find a minimum Hamiltonian cycle that traverses each arc (s_i, t_i) in the specified direction from s_i to t_i, where we identify the weight of the arc (s_i, t_i) with that of the corresponding edge $\{s_i, t_i\}$.

If the objective "Hamiltonian cycle" is substituted by "Hamiltonian path" in the definition of the SCP, we get the path version of the stacker crane problem, and we call it the path stacker crane problem (PSCP).

The following results are important for the discussion in Sects. 3 and 4.

Lemma 1 *Diestel (2017) A connected (multi)graph has a directed Eulerian trail if and only if it has either 0 or 2 vertices of odd degree.*

By a simple deduction, we get the following result similar to that in Hong et al. (2014).

Lemma 2 *A connected directed (multi)graph has a directed Eulerian trail whenever either of the following two conditions holds. (i) each vertex has the same in-degree and out-degree. (ii) the in-degree of one vertex is equal to the out-degree of this vertex plus one, the out-degree of another vertex is equal to the in-degree of this vertex plus one, and the out-degree of other vertices is equal to the in-degree of them.*

Lemma 3 *Kawasaki and Takazawa (2020) For the PTSP with u_1 and u_2 being the two given endpoints, there exists a polynomial-time approximation algorithm that finds Hamiltonian paths S_1 and S_2 from u_1 to u_2 such that $w(S_1) \leq 2OPT' - w\{u_1, u_2\}$, $w(S_2) \leq \frac{3}{2}OPT'$, where OPT' denotes the weight of an optimal solution of the problem.*

3 Approximation algorithms for the PSCP

In this section, we give two polynomial-time algorithms for the PSCP which will be used when we design the algorithm for the CPTSP in next section.

In the PSCP, we are given a mixed multigraph $G' = (V', E', D)$, where $V' = \{s_i, t_i \mid i \in [k]\}$, (V', E') is an undirected complete graph with an edge weight function w satisfying the triangle inequality, and $D = \{(s_i, t_i) \mid i \in [k]\}$. The objective is to find a minimum Hamiltonian path from s_1 to t_k that traverses each arc (s_i, t_i) in the specified direction from s_i to t_i. The first algorithm for the PSCP is as follows.

Algorithm 1

Input: A mixed multigraph $G' = (V', E', D)$, where $V' = \{s_i, t_i \mid i \in [k]\}$, (V', E') is an undirected complete graph, and $D = \{(s_i, t_i) \mid i \in [k]\}$. An edge weight function $w : E' \to \mathbb{R}_+$ satisfying the triangle inequality.

Output: Path P_{PSCP1}.

Begin

Step 1: Find a minimum bipartite matching between the head set $T = \{t_1, t_2, \cdots, t_{k-1}\}$ and the tail set $S = \{s_2, s_3, \cdots, s_k\}$.

Step 2: Initialize E_1 to be empty. For each edge included in the above matching, associate a direction with it, going from T to S, and insert it into E_1. (This results in $m \geq 1$ disjoint connected components, each of which consists of edges with the associated directions in the matching and arcs in D, and we denote these m disjoint connected components by $R_i, i \in [m]$.)

Step 3: Condense each R_i into a single vertex n_i. Define

$$d\{n_i, n_j\} = \min\{w\{u, v\} \mid u \in {R_i}', v \in {R_j}'\},$$

where ${R_i}'$ represents the set of all vertices in R_i, except for s_1 and t_k.

Step 4: Find a minimum spanning tree for the vertices in $\{n_i \mid i \in [m]\}$, where the minimum is with respect to the distance function d defined at Step 3.

Step 5: To begin with, make two copies of each edge in the spanning tree. Then, associate one direction with one copy, and the opposite direction with the other. At last, insert these edges with the associated directions into E_1. (This results in a directed graph $G'_1 = (V', E_1 \bigcup D)$.)

Step 6: Find a directed Eulerian trail from s_1 to t_k in G'_1.

Step 7: Using the triangle inequality, we get a Hamiltonian path P_{PSCP1} from s_1 to t_k traversing each arc (s_i, t_i) in the specified direction from s_i to t_i.

End

Example 1 Assume $G' = (V', E', D)$ with (V', E') being an undirected complete graph, $V' = \{s_i, t_i \mid i \in [4]\}$, and $D = \{(s_i, t_i) \mid i \in [4]\}$. The weight of edges is as follows: $w\{s_1, t_1\} = 1.6$, $w\{s_1, t_2\} = 1.5$, $w\{s_1, t_3\} = 1.4$, $w\{s_1, t_4\} = 1.3$, $w\{s_1, s_2\} = 1.2$, $w\{s_1, s_3\} = 1.5$, $w\{s_1, s_4\} = 1.4$, $w\{s_2, t_1\} = 1.5$, $w\{s_2, t_2\} = 1.9$, $w\{s_2, t_3\} = 1.8$, $w\{s_2, t_4\} = 1.6$, $w\{s_2, s_3\} = 1.1$, $w\{s_2, s_4\} = 1.2$, $w\{s_3, t_1\} = 1.6$, $w\{s_3, t_2\} = 1.9$, $w\{s_3, t_3\} = 1.8$, $w\{s_3, t_4\} = 1.7$, $w\{s_3, s_4\} = 2$, $w\{s_4, t_1\} = 1.5$, $w\{s_4, t_2\} = 1.7$, $w\{s_4, t_3\} = 2$, $w\{s_4, t_4\} = 1.7$, $w\{t_1, t_2\} = 1.2$, $w\{t_1, t_3\} = 1.7$, $w\{t_1, t_4\} = 1.2$, $w\{t_2, t_3\} = 1.8$, $w\{t_2, t_4\} = 1.3$, $w\{t_3, t_4\} = 1.4$.

At the beginning of our algorithm, there are four arcs, $\{(s_i, t_i) \mid i \in [4]\}$, $T = \{t_1, t_2, t_3\}$, and $S = \{s_2, s_3, s_4\}$ (seeing Fig. 1a). Find a minimum bipartite matching between T and S. By Step 2 in Algorithm 1, we get two connected components (seeing Fig. 1b). Condense each connected component into a single vertex, and by using the distance function d, we find a minimum spanning tree for vertices n_1, n_2, i.e., the edge $\{s_2, s_3\}$ (seeing Fig. 1c). Make two copies of the edge $\{s_2, s_3\}$ with opposite directions: one is (s_2, s_3) and the other is (s_3, s_2). Then we find a directed Eulerian trail $(s_1, (s_1, t_1), t_1, (t_1, s_2), s_2, (s_2, s_3), s_3, (s_3, t_3), t_3, (t_3, s_3), s_3, (s_3, s_2), s_2, (s_2, t_2), t_2, (t_2, s_4), s_4, (s_4, t_4), t_4)$ in the graph G'_1 with vertex set $V' = \{s_i, t_i \mid i \in [4]\}$ and directed edge set $\{(s_1, t_1), (t_1, s_2), (s_2, s_3), (s_3, t_3), (t_3, s_3), (s_3, s_2), (s_2, t_2), (t_2, s_4), (s_4, t_4)\}$ (seeing Fig. 1d). Using the triangle inequality, we get a path $(s_1, (s_1, t_1), t_1, (t_1, s_3), s_3, (s_3, t_3), t_3, (t_3, s_2), s_2, (s_2, t_2), t_2, (t_2, s_4), s_4, (s_4, t_4), t_4)$ (seeing Fig. 1e).

Lemma 4 *Algorithm 1 outputs a Hamiltonian path P_{PSCP1} with weight at most $3OPT'' - 2U$, where OPT'' denotes the weight of an optimal solution of the problem and $U = \sum_{i=1}^{k} w(s_i, t_i)$.*

We first show that P_{PSCP1} is a Hamiltonian path from s_1 to t_k that traverses each arc (s_i, t_i) in the specified direction from s_i to t_i. According to the directions of the edges of the bipartite matching at Step 2, we know that, except for s_1 and t_k, the in-degree and out-degree of each vertex in these connected components are equal. The edges of the spanning tree created at Step 4 connect these disjoint connected components produced at Step 2 into one. Since at Step 5 for each edge of the spanning tree, we add two edges with opposite directions into E_1, the in-degree and out-degree of each vertex are still equal, except for s_1 and t_k. Step 1 and Step 3 of the algorithm guarantee that s_1 and t_k are the only two vertices with odd degree. For specific, the in-degree of $s_1(t_k)$ is 0(1) and the out-degree of $s_1(t_k)$ is 1(0). At Step 6, by Lemma 2, we get the Eulerian trail from s_1 to t_k in the directed graph $G'_1 = (V', E_1 \bigcup D)$. At Step 7, using the triangle inequality, we get a desired Hamiltonian path.

In the following, we consider the weight of P_{PSCP1}. Since an optimal path of this problem contains a bipartite matching between T and S and all arcs in D, its weight can not be smaller than that of the minimum bipartite matching obtained at Step 1 and the arcs in D. For convenience, we denote the weight of the minimum bipartite matching by M. Then $M + U \leq OPT''$. Since the edges with the associated directions in the optimal path, except for the arcs in D, can connect these disjoint connected components

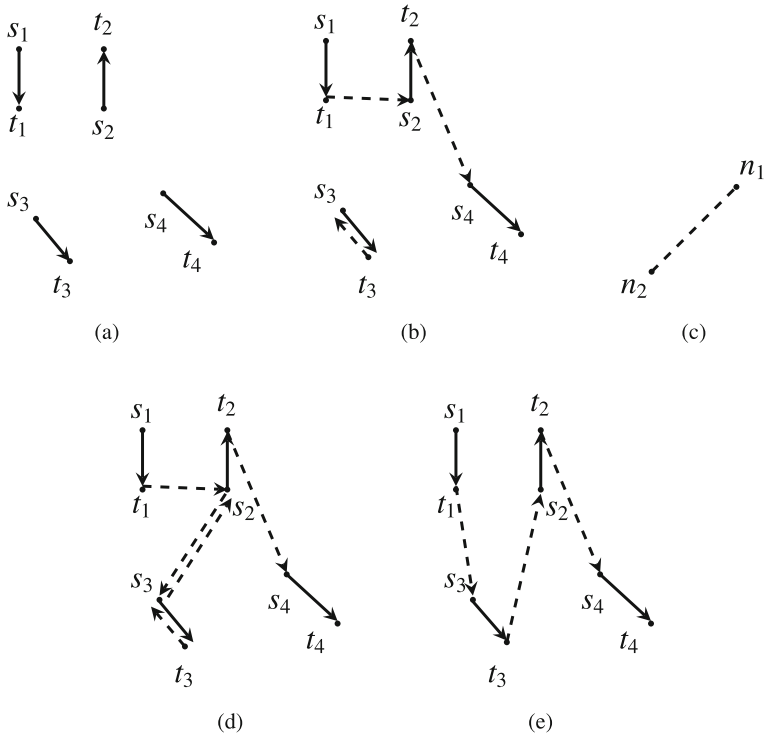

Fig. 1 Algorithm 1 applied to a graph

into one and the spanning tree created at Step 4 also connects these disjoint connected components into one, the weight of the minimum spanning tree at Step 4 must be no greater than $OPT'' - U$. Therefore, the weight of the directed Eulerian trail at Step 6 is at most $M + U + 2(OPT'' - U)$, and then is at most $3OPT'' - 2U$. Using the triangle inequality, we get the weight of the path P_{PSCP1} is at most the weight of the directed Eulerian trail, as desired.

The following is the second algorithm for the PSCP.

The following is an example to show the process of Algorithm 2.

Example 2 Assume $G'' = (V'', E'', D)$ with (V'', E'') being an undirected complete graph, $V'' = \{s_i, t_i \mid i \in [4]\}$, and $D = \{(s_i, t_i) \mid i \in [4]\}$. The weight of edges is as follows: $w\{s_1, t_1\} = 2.2, w\{s_1, t_2\} = 1.2, w\{s_1, t_3\} = 1.3, w\{s_1, t_4\} = 1.2, w\{s_1, s_2\} = 1.4, w\{s_1, s_3\} = 1.6, w\{s_1, s_4\} = 1.7, w\{s_2, t_1\} = 1.5, w\{s_2, t_2\} = 1.5, w\{s_2, t_3\} = 1.4, w\{s_2, t_4\} = 1.4, w\{s_2, s_3\} = 1.3, w\{s_2, s_4\} = 1.2, w\{s_3, t_1\} = 2, w\{s_3, t_2\} = 1, w\{s_3, t_3\} = 1.8, w\{s_3, t_4\} = 2, w\{s_3, s_4\} = 2.4, w\{s_4, t_1\} = 1.8, w\{s_4, t_2\} = 1.5, w\{s_4, t_3\} = 1.8, w\{s_4, t_4\} = 2.5, w\{t_1, t_2\} = 1.4, w\{t_1, t_3\} = 2.6, w\{t_1, t_4\} = 1.7, w\{t_2, t_3\} = 2.2, w\{t_2, t_4\} = 2, w\{t_3, t_4\} = 2.1$. With a similar discussion with Example 1, we get a desired Hamiltonian path $(s_1, (s_1, t_1), t_1, (t_1, s_3), s_3, (s_3, t_3), t_3, (t_3, s_2), s_2, (s_2, t_2), t_2, (t_2, s_4), s_4, (s_4, t_4), t_4$ (seeing Fig. 2).

Algorithm 2

Input: A mixed multigraph $G' = (V', E', D)$, where $V' = \{s_i, t_i \mid i \in [k]\}$, (V', E') is an undirected complete graph and $D = \{(s_i, t_i) \mid i \in [k]\}$. An edge weight function $w : E' \to \mathbb{R}_+$ satisfying the triangle inequality.

Output: Path P_{PSCP2}.

Begin

Step 1: Condense arc (s_i, t_i) into a vertex n_i for each $i \in [k]$. For each pair i, j with $i, j \in [k]$, define

$$d\{n_i, n_j\} = \begin{cases} \min\{w\{s_i, s_j\}, w\{s_i, t_j\}, w\{t_i, s_j\}, w\{t_i, t_j\}\}, & \text{if } s_1, t_k \notin \{s_i, s_j, t_i, t_j\}, \\ \min\{w\{t_i, s_j\}, w\{t_i, t_j\}\}, & \text{if } s_1 = s_i, \\ \min\{w\{s_i, t_j\}, w\{t_i, t_j\}\}, & \text{if } s_1 = s_j, \\ \min\{w\{s_i, t_j\}, w\{s_i, s_j\}\}, & \text{if } t_k = t_i, \\ \min\{w\{t_i, s_j\}, w\{s_i, s_j\}\}, & \text{if } t_k = t_j, \\ w\{t_i, s_j\}, & \text{if } s_1 = s_i, t_k = t_j, \\ w\{s_i, t_j\}, & \text{if } s_1 = s_j, t_k = t_i. \end{cases}$$

.

Step 2: Find a minimum spanning tree for the vertices in $\{n_i \mid i \in [k]\}$, where the minimum is with respect to the distance function d defined at Step 1.

Step 3: Initialize E_1 to be empty. Insert all edges in the above spanning tree into E_1. Replace the vertex n_i with the arc (s_i, t_i) for each $i \in [k]$. (This results in a mixed graph with vertex set V', undirected edge set E_1 and directed edge set D.)

Step 4: Identify vertices with odd degree in the mixed graph obtained at Step 3. Then find a minimum perfect matching for all these vertices with odd degree, except for s_1 and t_k.

Step 5: Insert all edges in the above matching into E_1. (This also results in a mixed graph with vertex set V', undirected edge set E_1 and directed edge set D.)

Step 6: Find an Eulerian trail from s_1 to t_k in the base graph of mixed graph obtained at Step 5, i.e., ignoring the directions of the arcs in D.

Step 7: Associate each edge in E_1 with the direction of the Eulerian trail. For each arc in D that is incorrectly traversed, add two directed edges to E_1, both with direction opposite to that of the arc. (This results in a directed graph $G'_2 = (V', E_1 \bigcup D)$.)

Step 8: Find a directed Eulerian trail from s_1 to t_k in G'_2.

Step 9: Using the triangle inequality, we get a Hamiltonian path P_{PSCP2} from s_1 to t_k traversing each arc (s_i, t_i) in the specified direction from s_i to t_i.

End

Lemma 5 *Algorithm 2 outputs a Hamiltonian path P_{PSCP2} with weight at most $2OPT'' + 2U$, where OPT'' denotes the weight of an optimal solution of the problem and $U = \sum_{i=1}^{k} w(s_i, t_i)$.*

With a similar discussion with Lemma 4, after completion of Step 5, all vertices are of even degree, except for s_1 and t_k. The degrees of s_1 and t_k are both 1. This is obtained by the definition of d at Step 1 and by ignoring these two odd degree vertices s_1 and t_k at Step 4. According to Lemma 1, we find an Eulerian trail from s_1 to t_k in the base graph of mixed graph obtained at Step 5. Step 7 shows how to augment the mixed graph to allow the Eulerian trail to traverse arcs in D in the proper direction. Similarly, the definition of d at Step 1 and omission of these two odd degree vertices s_1 and t_k at Step 4 ensure that (s_1, t_1) and (s_k, t_k) are correctly traversed. When there exists an arc in D that is incorrectly traversed, add two directed edges to E_1, both with direction opposite to that of the arc, the in-degree and out-degree of each vertex are equal, except for s_1 and t_k. The in-degree of $s_1(t_k)$ is 0(1) and the out-degree of $s_1(t_k)$

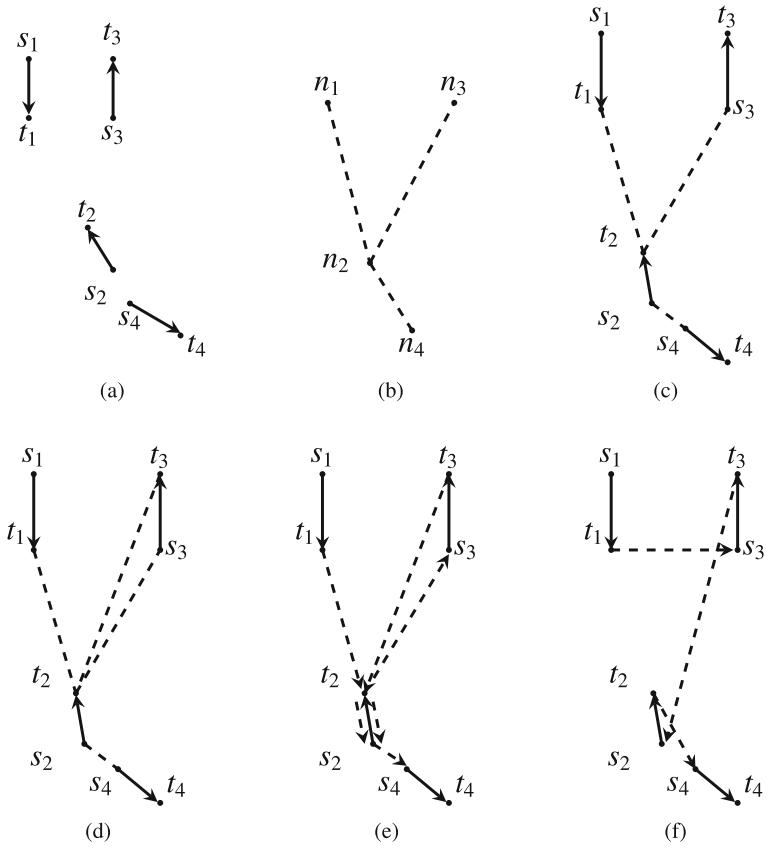

Fig. 2 Algorithm 2 applied to a graph

is 1(0). Then we get the directed Eulerian trail from s_1 to t_k traversing each arc (s_i, t_i) in the specified direction from s_i to t_i in the directed graph $G'_2 = (V', E_1 \bigcup D)$. At Step 9, using the triangle inequality, we get a desired Hamiltonian path.

In the following, we consider the weight of P_{PSCP2}. Since the edges with the associated directions in the optimal path, except for the arcs in D, can connect the vertices in $\{n_i \mid i \in [k]\}$, the weight of the spanning tree at Step 2 is at most $OPT'' - U$. Then the weight of the mixed graph obtained at Step 3 is at most OPT''. So the minimum perfect matching weight on these vertices of odd degree , except for s_1 and t_k, is at most OPT''. Therefore the weight of the mixed graph obtained at Step 5 is at most $2OPT''$, and so the weight of the Eulerian trail at Step 6 is at most $2OPT''$. Note that the weight of all directed edges added at Step 7 is at most $2U$. Then the weight of the directed Eulerian trail at Step 8 is at most $2OPT'' + 2U$. Also using the triangle inequality, we get the weight of the Hamiltonian path P_{PSCP2} is at most $2OPT'' + 2U$, as desired.

Remark 1 The outputs of the above two algorithms both have weight related to the weight U of the arcs in D. Each of Algorithm 1 and Algorithm 2 is run, and the value

U relative to OPT'' will determine which algorithm we will choose. For specific, if $U \geq \frac{1}{4} OPT''$, we will choose Algorithm 1. Otherwise, choose the other one instead.

Lemma 6 *The time complexity of Algorithm 1 and Algorithm 2 is* $O(\max\{|V'|^3, |E'| \log\log |V'|\})$ *and* $O(\max\{|V'|^2 |E'|, |E'| \log\log |V'|\})$, *where* V' *is the set of vertices of a given mixed graph and* E' *is the set of undirected edges.*

The running time of Algorithm 1 depends on the running time of finding the minimum bipartite matching and the minimum spanning tree. According to Grinman (2015), the running time of the Hungarian algorithm for finding the minimum bipartite matching is $O(|V'|^3)$. According to Yao (1975), the running time of the minimum spanning tree algorithm is $O(|E'| \log\log |V'|)$.

The running time of Algorithm 2 depends on the running time of finding the minimum perfect matching and the minimum spanning tree. There is a Blossom algorithm for computing minimum perfect matching by Edmonds (1965). The running time of the Blossom algorithm for finding the minimum perfect matching is $O(|V'|^2 |E'|)$.

4 Approximation algorithm for the CPTSP

In this section, we first design an approximation algorithm for the CPTSP, then we analyze its approximation ratio.

Recall that in the CPTSP, the start-vertex and end-vertex are both specified for each cluster. In order to apply the algorithms designed in Sect. 3 for the PSCP problem, we assume that the start-vertex (end-vertex) is $s_i(t_i)$ for each V_i, $i \in [k]$. Obviously, s_1 coincides with s and t_k coincides with t. Our algorithm mainly consists of four steps. At Step 1, for each fixed $i \in [k]$, we find the $path_i$, a path from s_i to t_i that goes through all vertices in V_i, and this is exactly the PTSP with given start-vertex and end-vertex. At Step 2, we replace the $path_i$ by the arc (s_i, t_i). At Step 3, we apply Algorithm 1 and Algorithm 2 to find a Hamiltonian path P_{PSCP} from s_1 to t_k that traverses each arc (s_i, t_i). At Step 4, we replace the arc (s_i, t_i) by the path $path_i$ obtained at Step 1.

Algorithm 3

Input: A complete graph $G = (V, E)$ with an edge weight function $w : E \to \mathbb{R}_+$, clusters $V_1, \cdots, V_k$, and the start-vertex (end-vertex) $s_i(t_i)$ for each V_i, $i \in [k]$.
Output: Hamiltonian path T.
Begin
Step 1: For each V_i, $i \in [k]$, compute $path_i$, a Hamiltonian path with start-vertex s_i and end-vertex t_i.
Step 2: Replace the $path_i$ with the arc (s_i, t_i). (This results in a mixed multigraph with vertex set $V' = \{s_i, t_i \mid i \in [k]\}$, V' and undirected edge set form a complete graph and $D = \{(s_i, t_i) \mid i \in [k]\}$.)
Step 3: Apply Algorithm 1 and Algorithm 2 to find the solution P_{PSCP} of PSCP with less weight in the mixed multigraph obtained above.
Step 4: In P_{PSCP}, replace the arc (s_i, t_i) by the $path_i$, $i \in [k]$.
End

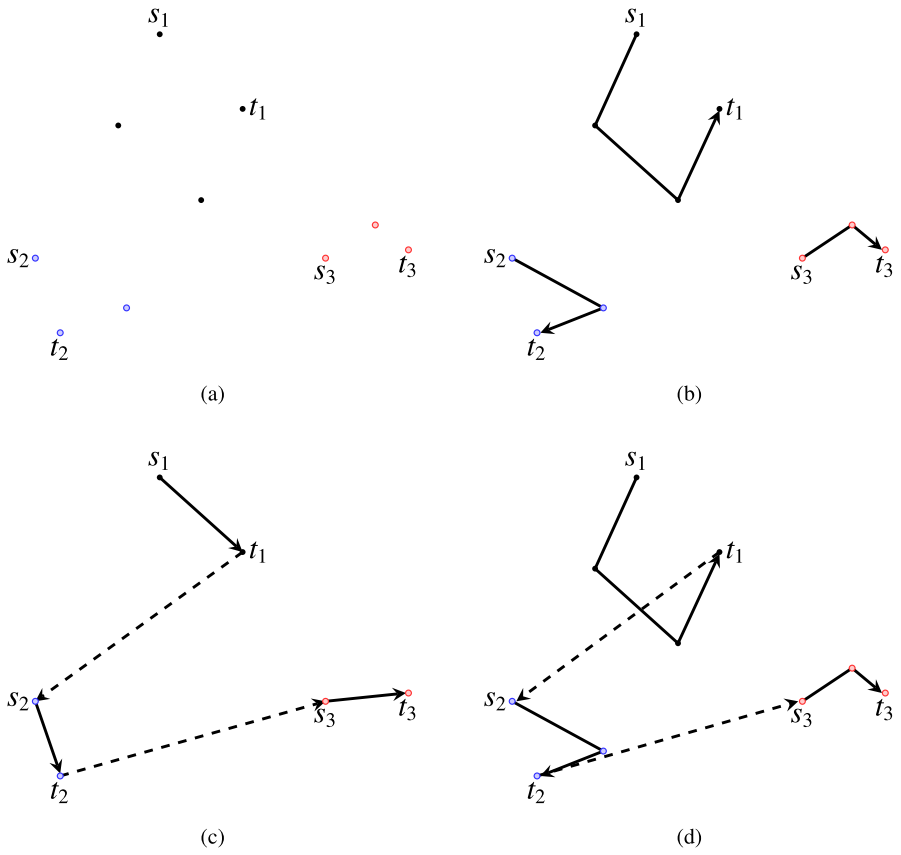

Fig. 3 Illustration of Algorithm 3

In this algorithm, we are involved in the PTSP and the PSCP that are both solvable in polynomial time, so our algorithm runs in polynomial time.

Example 3 See this example for a sample execution of the algorithm.

In this example, we are given three clusters (seeing Fig. 3a). We first compute $path_i$ with specified start-vertex s_i and end-vertex t_i in each cluster V_i, $i \in [k]$ (seeing Fig. 3b). Then we replace the $path_i$ with the arc (s_i, t_i) and solve the PSCP (seeing Fig. 3c). Using Algorithm 1 and Algorithm 2 to get the solution of PSCP, then we replace the arc by the $path_i$ in each cluster (seeing Fig. 3d).

Let OPT denote both an optimal solution of the CPTSP and its total weight. We get the approximation ratio of Algorithm 3 as follows.

Theorem 1 *Let T be the path output by Algorithm 3. Then $w(T) \leq \frac{8}{3} OPT$.*

The algorithm consists of two subproblems: the PTSP with given start-vertex and end-vertex, and the PSCP. Denote by P_i the induced path of the OPT on V_i for each $i \in [k]$. Let $L = \sum_{i\in[k]} \sum_{e\in P_i \cap OPT} w(e)$ and $L' = OPT - L$. Recall that

$U = \sum_{i=1}^{k} w(s_i, t_i)$. By Lemma 3, we get $w(\sum_{i=1}^{k} path_i) \leq \min(2L - U, \frac{3}{2}L)$. Therefore, $w(\sum_{i=1}^{k} path_i) \leq \min(2\,L - U, \frac{3}{2}L) \leq 2\,L - U$.

Note that there exists a solution to the PSCP with weight $L^{'} + U$. By Lemmas 4 and 5, the weight of the two solutions returned by Algorithm 1 and Algorithm 2 is at most $3\,L^{'} + U$ and $2\,L^{'} + 4U$, respectively. Therefore, using the fact that for a set of quantities, the minimum is less than or equal to any convex combination of them, we get

$$\begin{aligned} w(P_{\text{PSCP}}) &\leq \min(3L^{'} + U, 2L^{'} + 4U) \\ &\leq \frac{2}{3}(3L^{'} + U) + \frac{1}{3}(2L^{'} + 4U) \\ &= \frac{8}{3}L^{'} + 2U. \end{aligned}$$

At Step 4 of Algorithm 3, by replacing the arc (s_i, t_i) by $path_i$ for each $i \in [k]$, we obtain

$$\begin{aligned} w(T) &= w(\sum_{i=1}^{k} path_i) - U + w(P_{\text{PSCP}}) \\ &\leq (2L - U) - U + (\frac{8}{3}L^{'} + 2U) \\ &= 2L + \frac{8}{3}L^{'} \\ &\leq \frac{8}{3}(L + L^{'}) = \frac{8}{3}OPT. \end{aligned}$$

5 Discussion

In this paper, we only consider the case that the start-vertex and the end-vertex are both given in each cluster. Other cases including two endpoints in each cluster are given but we are free to choose the start-vertex, only the start-vertex is given in each cluster, and neither of the endpoints is given in each cluster, are also interesting problems to consider.

Acknowledgements The authors would like to thank the referees for giving this paper a careful reading and many valuable comments and suggestions.

Funding This work was supported by the NSF of China (No. 11971146), the NSF of Hebei Province of China (Nos. A2019205089, A2019205092), Overseas Expertise Introduction Program of Hebei Auspices (25305008) and the Graduate Innovation Grant Program of Hebei Normal University (No. CXZZSS2022052).

Data Availability Enquiries about data availability should be directed to the authors.

Declarations

Declarations The authors have not disclosed any competing interests.

References

An HC, Kleinberg R, Shmoys DB (2015) Improving Christofides algorithm for the s-t path TSP. J ACM 62(5):1–28
Anily S, Bramel J, Hertz A (1999) A 5/3-approximation algorithm for the clustered traveling salesman tour and path problems. Oper Res Lett 24:29–35
Arkin EM, Hassin R, Klein L (1994) Restricted delivery problems on a network. Networks 29:205–216
Bland RG, Shallcross DF (1989) Large travelling salesman problems arising from experiments in X-ray crystallography: a preliminary report on computation. Oper Res Lett 8(3):125–128
Chisman JA (1975) The clustered traveling salesman problem. Comput Oper Res 2:115–119
Christofides N (1976) Worst-case analysis of a new heuristic for the Travelling Salesman Problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University
Diestel R (2017) Graph theory. Springer, New York
Edmonds J (1965) Paths, trees and flowers. Can J Math 17:449–467
Frederickson GN, Hecht MS, Kim CE (1978) Approximation algorithms for some routing problems. SIAM J Comput 7(2):178–193
Gendreau M, Hertz A, Laporte G (1996) The traveling salesman problem with backhauls. Comput Oper Res 23:501–508
Gottschalk C, Vygen J (2018) Better s-t-tours by Gao trees. Math Program 172:191–207
Grinman A (2015) The Hungarian algorithm for weighted bipartite graphs. Seminar in Theoretical Computer Science
Grötschel M, Holland O (1991) Solution of large-scale symmetric traveling salesman problems. Math Program 51:141–202
Guttmann-Beck N, Hassin R, Khuller S, Raghavachari B (2000) Approximation algorithms with bounded performance guarantees for the clustered traveling salesman problem. Algorithmica 28:422–437
Hoogeveen JA (1991) Analysis of Christofides heuristic: some paths are more difficult than cycles. Oper Res Lett 10:291–295
Hong YM, Lai HJ, Liu QH (2014) Supereulerian digraphs. Discret Math 330:87–95
Jacobson EU, Argon NT, Ziya S (2012) Priority assignment in emergency response. Oper Res 60(4):813–832
Jongens K, Volgenant T (1985) The symmetric clustered traveling salesman problem. Eur J Oper Res 19:68–75
Kawasaki M, Takazawa T (2020) Improving approximation ratios for the clustered travelling salesman problem. J Oper Res Soc Jpn 63(2):60–70
Plante RD, Lowe TJ, Chandrasekaran R (1987) The product matrix travelling salesman problem: An Application and Solution Heuristics. Oper Res 35:772–783
Sebő A, van Zuylen A (2016) The salesmans improved paths: A 3/2+1/34 approximation. In: Proceedings of 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp 118–127
Traub V, Vygen J (2019) Approaching 3/2 for the s-t path TSP. J ACM 66(2):1–17
Yang X, Feng L (2013) Inventory routing problem: routing and scheduling approach with the objective of slack maximization. Transp Res Rec 2378(1):32–42
Yao A (1975) An O($|E|\log\log|V|$) algorithm for finding minimum spanning trees. Inform Process Lett 4:21–23
Zenklusen R (2019) A 1.5-approximation for path TSP. In: Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms, pp 1539–1549
Zhu L, Gong Y, Xu Y, Gu J (2019) Emergency relief routing models for injured victims considering equity and priority. Ann Oper Res 283:1573–1606

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.