An Improved Approximation Algorithm for the Subpath Planning Problem and Its Generalization

Hanna Sumita[†], **Yuma Yonebayashi**[‡], **Naonori Kakimura**[§], **Ken-ichi Kawarabayashi**[†]
[†]National Institute of Informatics
[‡]University of Tokyo
[§]Keio University
{sumita, k_keniti}@nii.ac.jp, yuuma_yonebayashi@mist.i.u-tokyo.ac.jp, kakimura@math.keio.ac.jp

Abstract

This paper focuses on a generalization of the traveling salesman problem (TSP), called the *subpath planning problem* (SPP). Given $2n$ vertices and n independent edges on a metric space, we aim to find a shortest tour that contains all the edges. SPP is one of the fundamental problems in both artificial intelligence and robotics. Our main result is to design a 1.5-approximation algorithm that runs in polynomial time, improving the currently best approximation algorithm. The idea is direct use of techniques developed for TSP. In addition, we propose a generalization of SPP called the *subgroup planning problem* (SGPP). In this problem, we are given a set of disjoint groups of vertices, and we aim to find a shortest tour such that all the vertices in each group are traversed sequentially. We propose a 3-approximation algorithm for SGPP. We also conduct numerical experiments. Compared with previous algorithms, our algorithms improve the solution quality by more than 10% for large instances with more than 10,000 vertices.

1 Introduction

The *Subpath Planning Problem* (SPP) is one of the fundamental problems in the artificial intelligence (AI) research community, as well as in the robotics research community. SPP can be formulated as follows: Given n independent edges (subpaths) in a complete graph on $2n$ vertices with length (weights) assigned to edges, we need to find a shortest tour that travels all given subpaths. SPP has widespread applications in those areas, such as a polishing robot [Tong-ying *et al.*, 2004] and electronic printings [Gyorfi *et al.*, 2010]. For example, a "polishing" robot needs to polish nicks on a surface of a car. We can regard the nicks as subpaths in the 2-dimensional plane, like Figure 1 (a). To reduce the operation cost, we need to design a shortest route that goes through the given three paths as shown in Figure 1 (b). For other applications, see e.g., [Safilian *et al.*, 2016] and references therein.

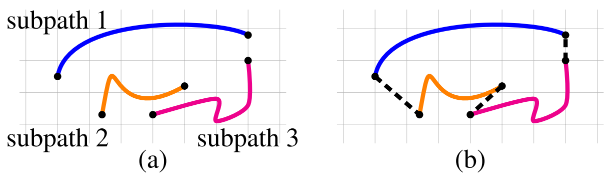

Figure 1: An SPP instance with $n = 3$.

SPP is a (strict) generalization of the well-known problem, the *traveling salesman problem* (TSP). Indeed, when every subpath is identical to one vertex and the lengths of subpaths are all zero, the corresponding problem of SPP is exactly TSP. TSP is the central combinatorial optimization problem and it has been studied extensively in both theory and applications. While TSP is NP-hard and inapproximable within any approximation ratio [Sahni and Gonzalez, 1976], some approximation algorithms are proposed for some special cases, such as the metric TSP [Christofides, 1976], the Euclidean TSP [Arora, 1998; Mitchell, 1999] and the graphic TSP [Mömke and Svensson, 2011; 2016][1]. Many practically efficient heuristics are also proposed; for example, 2-Opt, 3-Opt, the Lin-Kernighan algorithm [Lin and Kernighan, 1973] and bio-inspired algorithms (such as genetic algorithms and the artificial ant colony [Dorigo and Gambardella, 1997]).

Since SPP is at least as hard as TSP, the main focus is to develop efficient algorithms that find a nearly optimal solution. Hereafter, we assume that SPP is *metric*, i.e., the triangle inequality holds for any edge *except for the subpaths*. Meta-heuristics based on genetic algorithms are proposed in [Tong-ying *et al.*, 2004; Gyorfi *et al.*, 2010]. However, the algorithms have no theoretical guarantee on the solution quality, and not scalable as reported in [Safilian *et al.*, 2016]. The first scalable algorithm with theoretical approximation guarantee was presented by Safilian *et al.* [2016]. Their algorithm can find a 2-approximate solution[2] in $O(n^3)$ time, where n is the number of input subpaths.

In this paper, we provide the following result for SPP.

Theorem 1.1. *There exists a 1.5-approximation algorithm that runs in* $\mathrm{O}(n^3)$ *time for the metric SPP.*

The idea of our algorithm is to make direct use of

[1]In the graphic TSP, the metric is defined by the shortest path distance on a graph.

[2]A feasible solution is *α-approximate* if its objective value is at most α times the optimal value. An *α-approximation algorithm* is an algorithm that returns an α-approximate solution for any instance.

Christofides' techniques [Christofides, 1976] developed in the literature of TSP. Christofides' algorithm is applicable only for the metric TSP, but edge lengths in the underlying graph for the metric SPP may not satisfy the triangle inequalities, because given subpaths may be longer than the distances between the end vertices. The previous algorithm [Safilian *et al.*, 2016] constructs a TSP instance, which is nearly metric, from a given SPP instance, and then applies Christofides' techniques to obtain a 2-approximate solution. In order to make an obtained tour go through all the subpaths, their algorithm needs an additional step. This makes the approximation ratio worse than the ratio 1.5 (that is the case for the metric TSP). In contrast, we directly apply Christofides' techniques to a given SPP instance. The main technical challenge is to modify each step of Christofides' algorithm so that an obtained tour always goes through the subpaths. We prove that the adaption does not make the ratio worse, which leads to an 1.5-approximation algorithm for the metric SPP.

We further show that our analysis is tight in the sense that there is an instance of the metric SPP such that our algorithm returns a solution whose ratio is arbitrarily close to 1.5. Let us remark that Christofides' algorithm (for the metric TSP) is also known to be tight, and it has been a long-standing open problem to improve the approximation ratio 1.5 for the metric TSP. As mentioned above, SPP is a generalization of TSP, and thus improving the approximation ratio in Theorem 1.1 would also improve the approximation ratio 1.5 for the metric TSP. So the prospect is not at all bright to improve approximation ratio of Theorem 1.1.

In addition, we introduce a more general problem called the *subgroup planning problem* (SGPP). Instead of subpaths given in SPP, we are given pairwise disjoint groups of vertices. We need to find a shortest tour that travels all the vertices consecutively in each group. In Figure 2 (a), we are given three groups, and the purpose here is to make a tour that travels every group like Figure 2 (b). SGPP is a natural generalization of both TSP and SPP, as it coincides with TSP when all groups have size one, and with SPP when all groups have size two. SGPP can model some scheduling problems in which similar jobs are required to be processed at once; for example, controlling a robot arm to do different jobs, electronic printings with different types of holes, and so on. See also the later paragraph about related work. By extending our approximation algorithm in Theorem 1.1, we propose a $(1.5 + \alpha)$-approximation algorithm for the metric SGPP, i.e., the triangle inequality holds for any edge *outside groups*. Here α is an approximation ratio for the *traveling salesman path problem* (TSPP), i.e., the problem of finding a shortest path that starts from a vertex s and ends at another vertex t after traveling all vertices. Christofides' technique can be adapted to obtain an algorithm with $\alpha = 1.5$ [Hoogeveen, 1991]. When two vertices s and t are specified, TSPP admits only 1.566-approximation [Gottschalk and Vygen, 2016].

Theorem 1.2. *There exists a 3-approximation algorithm running in polynomial time for the metric SGPP.*

We also conduct numerical experiments on VLSI open datasets to show the superiority of our approximation algorithms over the state-of-art algorithm [Safilian *et al.*, 2016].

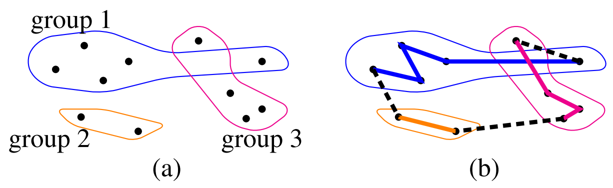

Figure 2: An SGPP instance with three groups.

Our approximation algorithms can solve instances with more than 10,000 vertices. The solution quality is improved by more than 10% for large instances.

Related Work SPP is a variant of path planning problems with spatial constraints, which have been extensively studied in various areas such as AI, robotics, computer games; see e.g., [Jaillet and Porta, 2013; Kapadia *et al.*, 2013; Nash *et al.*, 2009; Surynek, 2015]. We remark that SPP has a close connection to the *Rural Postman Problem* (RPP) [Frederickson, 1979]. RPP and its variants have applications related to those areas, such as street sweeping, snow plowing, and so on [Eiselt *et al.*, 1995]. Existing papers on the metric RPP assume that all edges satisfy the triangle inequality. On the other hand, this paper assumes a weaker property that the triangle inequality holds only for a subset of edges, demanded from AI applications. We also remark that the metric SGPP is a generalization of the metric *clustered TSP* (CTSP), weakening triangle inequality in a similar way. CTSP can also be used to applications arising in AI and robotics, such as automated warehouse routing [Chisman, 1975; Lokin, 1979] and production planning [Lokin, 1979].

Besides related work on TSP we have mentioned before, TSP and its related problems have been studied in the AI research community, due to widespread applications [Ducomman *et al.*, 2016; Kumar *et al.*, 2013; Tu *et al.*, 2010; Zhang and Looks, 2005]. For example, [Zhang and Looks, 2005] proposed a heuristic based on the Lin-Kernighan algorithm [Lin and Kernighan, 1973] by incorporating backbone information of TSP.

Organization Section 2 gives precise definitions of SPP and SGPP. Section 3 proposes the 1.5-approximation algorithm for SPP and shows that our analysis is tight. Extending this algorithm, Section 4 provides the $(1.5 + \alpha)$-approximation algorithm for SGPP. Section 5 shows the experimental results. Finally, Section 6 concludes this paper.

2 Preliminaries

In this section, we define terminologies and give precise definitions of SPP and SGPP. Let $G = (V, E)$ be an undirected graph. For any vertex subset U, we say that a path is *U-Hamiltonian* if it travels only vertices in U and visits every vertex in U exactly once. A *tour* on G is a cycle going though all the vertices.

2.1 The Subpath Planning Problem

Given a set V of $2n$ vertices, a set S of n independent edges on V, and edge lengths $c : V \times V \to \mathbb{R}_+$, the *subpath plan-*

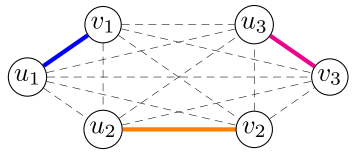

Figure 3: The underlying complete graph in an SPP instance. Each subpath is shown in a thick edge.

ning problem (SPP) is to find a tour $T = e_1, \ldots, e_{2n}$ in the complete graph $G = (V, E)$ on V such that T contains all edges in S, and the total length $\sum_{i=1}^{2n} c(e_i)$ of T is minimized. Each edge $e \in S$ is called a *subpath*. We denote the problem instance by (V, S, c). For any path $P = e'_1, \ldots, e'_\ell$, we write the total length of edges in P as $c(P) = \sum_{i=1}^{\ell} c(e'_i)$.

We say that (V, S, c) is *metric* if the triangle inequality $c(u, v) + c(v, w) \geq c(u, w)$ is satisfied for all $(u, v), (v, w), (u, w) \in E$ such that $(u, w) \in E \setminus S$. Note that the metric SPP is a generalization of the metric TSP.

We remark that the original subpath planning problem [Safilian *et al.*, 2016] was defined on the 2-dimensional plane with n subpaths $p_1, \ldots, p_n$. The problem falls into the metric SPP by defining $c(u, v)$ to be the length of p_i if u and v are the end vertices of p_i, and the distance between u and v otherwise. The triangle inequality $c(u, v)+c(v, w) \geq c(u, w)$ holds for any vertices u, v, w except for the case when u, w are the end vertices of some subpath, since its length may be longer than the distance between the end vertices. Thus the metric SPP generalizes this setting. For example, Figure 1 can be represented as the complete graph as in Figure 3.

2.2 The Subgroup Planning Problem

The subgroup planning problem is a generalization of SPP. Given a vertex set V, a set $\mathcal{S}$ of n pairwise disjoint vertex sets $S_1, \ldots, S_n$ with $\bigcup_i S_i = V$, and edge lengths $c : V \times V \to \mathbb{R}_+$, the *subgroup planning problem* (SGPP) is to find a tour T in the complete graph $G = (V, E)$ on V such that T contains an S_i-Hamiltonian path for all $i = 1, \ldots, n$, and the total length of T is minimized. Each vertex set in $\mathcal{S}$ is called a *subgroup*. Note that if the size of each subgroup is exactly two, then SGPP coincides with SPP. We denote the problem instance by $(V, \mathcal{S}, c)$.

In the complete graph $G = (V, E)$, we say that edges (u, v) with $u, v \in S_i$ $(i \in \{1, \ldots, n\})$ are *inner* edges, and the other edges are *outer* edges. Outer edges connect vertices in different subgroups. Similarly to SPP, we say that $(V, \mathcal{S}, c)$ is *metric* if the triangle inequality $c(u, v) + c(v, w) \geq c(u, w)$ is satisfied for any inner edges $(u, v), (v, w), (u, w)$ in each subgroup, and for any edges $(u, v), (v, w), (u, w)$ such that (u, w) is an outer edge.

3 Improved Algorithm for the Metric SPP

In this section, we describe an approximation algorithm for the metric SPP. Let (V, S, c) be an instance of the metric SPP. We denote by $G = (V, E)$ the complete graph on V.

Our algorithm is based on Christofides' algorithm for the metric TSP. So let us review it. We first find a minimum spanning tree R in the graph G, and then find a minimum-weight perfect matching M in the subgraph induced by the odd-degree vertices in R. Then each vertex in the subgraph $R \cup M$ has even degree, and hence we can construct an Eulerian cycle in $R \cup M$, i.e., a cycle C that uses every edge in $R \cup M$ exactly once. From this Eulerian cycle C, we create a tour by skipping visited vertices (shortcutting) as follows. Traverse C starting from an arbitrary edge, and define the order $(u_1, \ldots, u_{2n})$ of vertices in which they appear for the first time in C. Then set $T = (u_1, u_2), (u_2, u_3), \ldots, (u_{2n}, u_1)$. We see that the total length of R is at most the optimal value, and the total weight of M is at most a half of the optimal value. This guarantees that the obtained tour is 1.5-approximation. The triangle inequality is necessary to show that the shortcutting does not increase the tour length.

For the metric SPP, the difference from the metric TSP is in the constraint that we have to go though all the given subpaths, and that c does not necessarily satisfy the triangle inequality in the whole graph. We need to find a minimum spanning tree containing all the edges in S, and to shortcut edges while keeping the edges in S in the tour, without increasing the length.

For any edge set S', we say that a tree is *S'-spanning* if it is a spanning tree that contains all edges in S'. The following lemma shows that, if edges in S do not share the end vertices, then we can find a minimum S-spanning tree by contracting each edge in S.

Lemma 3.1. *Let $G = (V, E)$ be a connected graph on n vertices with edge weight $c : V \times V \to \mathbb{R}_+$, and let $S \subseteq E$ be a set of independent edges. There is an $\mathrm{O}(n^2)$ time algorithm to find a minimum S-spanning tree of G.*

Proof. We construct a graph $G' = (V', E')$ by contracting each edge in S to one vertex. The edge weights c' on E' are defined as the minimum weight among parallel edges when contracting edges. Note that $|V'| = |V| - |S|$.

It is not difficult to see that any S-spanning tree T on G corresponds to some spanning tree T' on G' by contraction. Moreover, $c'(T') \leq c(T) - c(S)$, and the equality holds when T is a minimum spanning tree. Conversely, for any spanning tree T' on G', we can obtain some S-spanning tree T on G by uncontraction (i.e., the reverse operation of contraction), since edges in S are independent. The weight of T satisfies that $c(T) = c'(T') + c(S)$.

Thus finding a minimum S-spanning tree on G is equivalent to finding a minimum spanning tree on G', which can be found in $\mathrm{O}(n^2)$ time by Prim's algorithm (see e.g., [Korte and Vygen, 2002] for the description). Hence we can obtain a minimum S-spanning tree for G in $O(n^2)$ time. $\square$

From an Eulerian cycle C, we create a tour T by skipping edges not in S to guarantee that $S \subseteq T$. Traverse C starting from an arbitrary edge. We write $C = e_1, \ldots, e_l$. Since $S \subseteq C$, we can define the order $(e_{i_1}, \ldots, e_{i_n})$ of edges in S such that they appear in this order. In our tour T, instead of going through the path $e_{i_j+1}, \ldots, e_{i_{j+1}-1}$ $(j \leq n)$, we take a shortcut by an edge e'_j connecting the end vertices of the path. Here we denote $i_{n+1} = i_1$. We will prove that each e'_j is indeed a shortcut. Then set $T = e_{i_1}, e'_1, e_{i_2}, e'_2, \ldots, e_{i_n}, e'_n$.

Our algorithm is described below. We remark that Steps 1 and 5 are uniquely designed for the metric SPP.

Algorithm 1

1. Find a minimum S-spanning tree R of G.
2. Let V' be the set of vertices having odd degree in R, and let G' be the subgraph induced by V'.
3. Find a minimum-weight perfect matching M in G' (weight for M is defined by $\sum_{e \in M} c(e)$).
4. Construct an Eulerian cycle C on $R \cup M$.
5. Traverse C starting from an arbitrary edge. Let e_{i_j} be the jth subpath appearing in C. Let e'_j be the edge connecting the end vertices of the path $e_{i_j+1}, \dots, e_{i_{j+1}-1}$. Output the tour $T = e_{i_1}, e'_1, e_{i_2}, e'_2, \dots, e_{i_n}, e'_n$.

We first estimate the running time. We can find R in $\mathrm{O}(n^2)$ time by Lemma 3.1. A minimum-weight perfect matching on a complete graph can be found in $\mathrm{O}(n^3)$ time by using, e.g., the algorithm proposed in [Gabow, 1990]. The Eulerian cycle C can be found in $\mathrm{O}(n)$ time since $|R \cup M| = \mathrm{O}(n)$. Since C has $\mathrm{O}(n)$ edges, we need $\mathrm{O}(n)$ time to construct the output T. Thus the running time is $\mathrm{O}(n^3)$.

We remark that the output T of Algorithm 1 is a tour on G that contains all edges in S. To prove Theorem 1.1, it remains to show the approximation ratio. The following observation follows because (V, S, c) is metric.

Lemma 3.2. *For any path $(u_1, u_2), \dots, (u_{k-1}, u_k)$ on G, if $(u_1, u_i) \notin S$ for any $i > 2$, then (u_1, u_k) is a shortcut, i.e., $c(u_1, u_k) \leq \sum_{i=1}^{k-1} c(u_i, u_{i+1})$ holds.*

Proof. By definition of the metric SPP, we have $c(u_1, u_i) + c(u_i, u_{i+1}) \geq c(u_1, u_{i+1})$ for $i = 2, \dots, k-1$. Combining all inequalities leads to $\sum_{i=1}^{k-1} c(u_i, u_{i+1}) \geq c(u_1, u_k)$. $\square$

Proof of Theorem 1.1. We prove the theorem by showing that Algorithm 1 outputs a 1.5-approximation solution to the metric SPP. Let T be the output of Algorithm 1, and let T^* be any optimal tour. We denote $\mathrm{OPT} = c(T^*)$.

We first observe that $c(R) \leq \mathrm{OPT}$. Indeed, any path obtained by removing an edge in $T^* \setminus S$ is also an S-spanning tree of G. Since R is a minimum one, $c(R)$ is at most the total length of the path obtained from T^*, which is at most OPT.

We now bound $c(M)$. Let T' be a shortest tour on G', where G' is a complete graph on V'. We can obtain two perfect matchings M_1, M_2 by selecting edges in T' alternatively. Since M is a minimum-weight perfect matching, we have $c(M) \leq \min\{c(M_1), c(M_2)\} \leq c(T')/2$.

Let $\hat{T}$ be a cycle obtained from T^* by skipping vertices not contained in V'. Since $S \subseteq T^*$, any path in T^* satisfies the condition in Lemma 3.2. Hence Lemma 3.2 implies that edges in $\hat{T} \setminus T^*$ are shortcuts, i.e., $c(\hat{T}) \leq c(T^*) = \mathrm{OPT}$. Moreover, because $\hat{T}$ is also a tour on G', we have $c(T') \leq c(\hat{T})$. Thus $c(M') \leq c(T')/2 \leq \mathrm{OPT}/2$, and hence it follows that $c(C) = c(R) + c(M) \leq \mathrm{OPT} + \mathrm{OPT}/2 \leq 1.5 \cdot \mathrm{OPT}$.

Since $S \subseteq T$ by construction of T, every edge $(u, v) \in T \setminus S$ is a shortcut of a path in C connecting u and v by applying Lemma 3.2 together with the fact that $S \subseteq C$. Thus $c(T) \leq c(C)$ holds, and this proves the theorem. $\square$

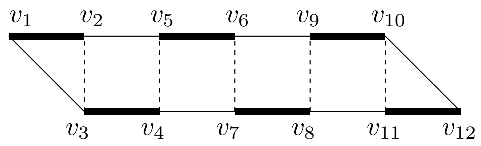

Figure 4: The bold edges denote edges in S, the dashed edges denote edges in E_1, and the other edges are in E_2.

3.1 Lower Bounds

In this subsection, we show that there exists an instance with positive subpath lengths such that our algorithm returns a solution whose approximation ratio is arbitrarily close to 1.5.

Theorem 3.3. *For any number $\varepsilon > 0$, there exists an instance such that Algorithm 1 returns a solution T with $c(T) \geq (1.5 - \varepsilon)\mathrm{OPT}$, where OPT denotes the optimal value.*

Proof. Let n be an even integer greater than or equal to $1/(2\varepsilon + 2\varepsilon^2)$. We first define a graph $G = (V, E)$ by $V = \{v_1, v_2, \dots, v_{2n-1}, v_{2n}\}$, $S = \{(v_{2i-1}, v_{2i}) \mid i = 1, \dots, n\}$, $E_1 = \{(v_{2i}, v_{2i+1}) \mid i = 1, \dots, n-1\}$, $E_2 = \{(v_{2i}, v_{2i+3}) \mid i = 0, \dots, n-1\}$, and $E = S \cup E_1 \cup E_2$, where we denote $v_0 = v_1$ and $v_{2n+1} = v_{2n}$ for convenience. Figure 4 illustrates the graph when $n = 6$. Note that S, E_1 and E_2 are pairwise disjoint. Define the weight of each edge $e \in E$ by $w(e) = 2 + \varepsilon$ for $e \in S$, 1 for $e \in E_1$, and $1 + \varepsilon$ for $e \in E_2$. Using the graph G, define c to be the shortest-path metric, i.e., $c(u, v)$ is the total weight of a shortest path between u and v in G where the edge weight is w. For example, $c(v_1, v_{12}) = 9 + 3\varepsilon$ in Figure 4.

We show that (V, S, c) satisfies the condition of the theorem. We observe that the subgraph $(V, S \cup E_1)$ is the unique minimum S-spanning tree, because it consists of S and minimum-weight edges. Since we have only two odd-degree vertices v_1 and v_{2n}, the minimum-weight matching is $(\{v_1, v_{2n}\})$. Since no shortcut occurs, our algorithm returns a tour consisting edges in $S \cup E_1 \cup (v_1, v_{2n})$, namely $(v_1, v_2), (v_2, v_3), \dots, (v_{2n}, v_1)$. The length of this tour is $c(S) + c(E_1) + c(v_1, v_{2n})$, which is equal to $n(2 + \varepsilon) + (n - 1) + \frac{n}{2}(1 + \varepsilon + 2 + \varepsilon) = (4.5n - 1) + 2n\varepsilon$. On the other hand, the optimal tour to the instance consists of edges in $S \cup E_2$. Its length is $c(S) + c(E_2) = n(2 + \varepsilon) + n(1 + \varepsilon) = (3 + 2\varepsilon)n$. Therefore, the ratio is $c(T)/\mathrm{OPT} = (4.5n - 1 + 2n\varepsilon)/(3 + 2\varepsilon)n \geq 1.5 - \varepsilon$, where the inequality follows from the fact that $n \geq 1/(2\varepsilon + 2\varepsilon^2)$. Thus the theorem holds. $\square$

4 Algorithm for the Metric SGPP

In this section, we propose an approximation algorithm for the metric SGPP. Let $(V, \mathcal{S}, c)$ be a metric SGPP instance.

One might think that we can easily extend the idea of Algorithm 1 for the metric SPP. Namely, we find a minimum spanning tree by contracting each subgroup S_i to one vertex, compute a minimum-weight matching on the odd-degree vertices, and create a tour by shortcutting. However, shortcutting would not work for the metric SGPP. Since the obtained Eulerian cycle may go through some subgroup many times, the obtained tour may count the lengths in the subgroup many times, and thus it is hard to bound the approximation ratio.

The difficulty of this approach comes from the fact that we have to traverse all vertices in a subgroup sequentially; once we enter into a subgroup, we need to choose another vertex to leave after traveling all vertices in the subgroup. Thus we need to decide two end vertices connecting other subgroups when designing a route inside a subgroup. In the case of SPP, since each subgroup has size two, the route is uniquely determined, and this issue does not occur. For SGPP, we resolve the issue by dividing our algorithm into two phases: (i) decide a route with two end vertices inside each subgroup S_i, and then (ii) solve an SPP instance defined by (i).

In the first phase, we use algorithms for TSPP; given an undirected graph $G = (V, E)$, TSPP is to find a shortest V-Hamiltonian path. Note that end vertices are not specified. Let α be an approximation ratio for TSPP. Our algorithm finds an α-approximate S_i-Hamiltonian path for each subgroup S_i. Let (u_i, v_i) be the end vertices of the path in S_i. Note that, in our algorithm, we can use any α-approximation algorithm for TSPP. For the metric TSPP, the current best ratio is $\alpha = 1.5$ [Hoogeveen, 1991]. When the subgroup size is a small constant, we can use the dynamic programming algorithm for TSP [Bellman, 1962; Held and Karp, 1962]. Note that we can convert TSPP into TSP by adding a new vertex v_0 and setting $c(v_0, v) = 0$ for all vertices v.

We then use our 1.5-approximation algorithm proposed in Section 3. We construct an SPP instance consisting of u_i and v_i for all i, and set the length of each subpath to be the original length $c(u_i, v_i)$. Note that we do not use the length of the shortest S_i-Hamiltonian path for (u_i, v_i) obtained in the first phase, as the length of subpaths.

Our algorithm for SGPP is summarized below.

Algorithm 2

1. For each subgroup S_i, find an α-approximate S_i-Hamiltonian path P_i and let (u_i, v_i) be the end vertices of P_i.
2. Apply Algorithm 1 to obtain a feasible tour T' for the SPP instance (V', S', c), where $V' = \{u_i, v_i \mid i = 1, \dots, n\}$ and $S' = \{(u_i, v_i) \mid i = 1, \dots, n\}$.
3. Construct a tour T by replacing each edge (u_i, v_i) in T' with the path P_i for $i = 1, \dots, n$.

Step 1 with $\alpha = 1.5$ can be implemented in polynomial time by, e.g., the algorithm in [Hoogeveen, 1991]. Step 2 takes $O(n^3)$ time. The construction of the output can be done in linear time when $\alpha = 1.5$. Thus, our algorithm runs in polynomial time. We also observe that Algorithm 2 outputs a feasible solution for SGPP. Then we prove Theorem 1.2.

Proof of Theorem 1.2. We prove that Algorithm 2 is 3-approximation algorithm. We denote by ALG the total length of the output T from Algorithm 2. Let T^* be an optimal tour of $(V, \mathcal{S}, c)$, and let $\mathrm{OPT} = c(T^*)$. The tour T^* consists of S_i-Hamiltonian paths P_i^* $(i = 1, \dots, n)$ and outer edges which connect subgroups. Let $\mathrm{OPT}_{\mathrm{out}}$ (respectively, $\mathrm{ALG}_{\mathrm{out}}$) denote the total length of outer edges in T^* (respectively, T). Note that $\mathrm{ALG} = \mathrm{ALG}_{\mathrm{out}} + \sum_{i=1}^{n} c(P_i)$ and $\mathrm{OPT} = \mathrm{OPT}_{\mathrm{out}} + \sum_{i=1}^{n} c(P_i^*)$.

Take an arbitrary index $i \in [n]$. Let $\hat{P}_i$ denote the shortest S_i-Hamiltonian path. It follows from construction of P_i that $c(P_i) \le \alpha \cdot c(\hat{P}_i)$. Moreover, we have $c(\hat{P}_i) \le c(P_i^*)$ by definition of $\hat{P}_i$. Then $c(P_i) \le \alpha \cdot c(P_i^*)$ holds for each i.

It remains to show that $\mathrm{ALG}_{\mathrm{out}} \le 1.5 \cdot \mathrm{OPT}$. Let $\hat{T}$ be an optimal tour for (V', S', c). We remark that (V', S', c) is an SPP instance obtained from $(V, \mathcal{S}, c)$ by deleting vertices not in V'. In particular, the edge length of (u_i, v_i) in (V', S', c) is the original one. Hence $c(\hat{T})$ is at most OPT. Since T' is the output of Algorithm 1, we have $c(T') \le 1.5 \cdot c(\hat{T})$. We construct T by just replacing edges (u_i, v_i) in T'. Thus we have $\mathrm{ALG}_{\mathrm{out}} \le c(T') \le 1.5 \cdot c(\hat{T}) \le 1.5 \cdot \mathrm{OPT}$.

Recall that $\sum_{i=1}^{n} c(P_i^*) = \mathrm{OPT} - \mathrm{OPT}_{\mathrm{out}}$. Therefore,

$$\begin{aligned}\mathrm{ALG} &= \mathrm{ALG}_{\mathrm{out}} + \textstyle\sum_{i=1}^{n} c(P_i) \\ &\le 1.5 \cdot \mathrm{OPT} + \alpha \textstyle\sum_{i=1}^{n} c(P_i^*) \le (1.5 + \alpha)\mathrm{OPT}.\end{aligned}$$

Thus the approximation ratio of Algorithm 2 is $(1.5+\alpha)$. This together with a polynomial-time α-approximation algorithm with $\alpha = 1.5$ for the metric TSPP completes the proof. □

5 Experiments

In this section, we show experimental results to demonstrate the effectiveness of the proposed algorithms. All experiments were conducted on a server with Intel Xeon E5-2680 v3 and 1TB memory. All algorithms were implemented in C++.

5.1 Results for SPP

As SPP has applications in electronic printings [Gyorfi *et al.*, 2010], we use VLSI datasets for TSP available online[3]. Each instance has the set of points in the 2-dimensional Euclidean plane, where the number of points ranges from 130 to 21,214. Since placements of subpaths might affect the performance, we define the set of subpaths on each instance by the following two ways: (a) find a minimum-weight perfect matching with respect to the distances, and (b) find a perfect matching randomly. The setting (a) means that the end vertices of a subpath are expected to be close and the length will be short. The setting (b) produces random instances. The lengths of a subpath (u, v) are set to be $X \cdot d(u, v)$, where d is the original distance between u and v and X is an integer drawn from $[2, 10]$ uniformly at random.

We implemented our 1.5-approximation algorithm and the existing algorithm proposed in [Safilian *et al.*, 2016]. The numerical experiments conduced in [Safilian *et al.*, 2016] report that their algorithm outperforms other heuristic algorithms of [Tong-ying *et al.*, 2004; Gyorfi *et al.*, 2010]. Thus we omit comparison with these heuristic algorithms here.

Figure 5 shows the solution quality of the two algorithms when subpaths are generated as (a). In the figure, the bars represent the objective values excluding the total length of the subpaths. We can see that our algorithm returns a better solution for all the instances. In fact, the solution quality is improved by more than 10% for large instances. The results for case (b) (Figure 6) show even a bigger improvement. This is because case (b) has longer subpaths, which makes the

[3]http://www.math.uwaterloo.ca/tsp/vlsi/index.html

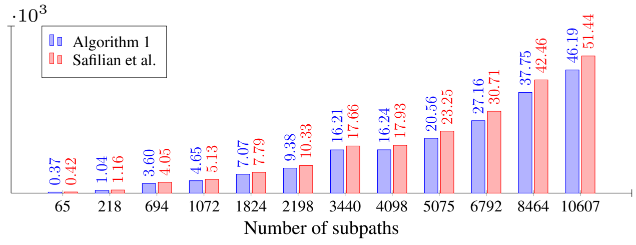

Figure 5: The length for SPP (excluding subpaths) (a).

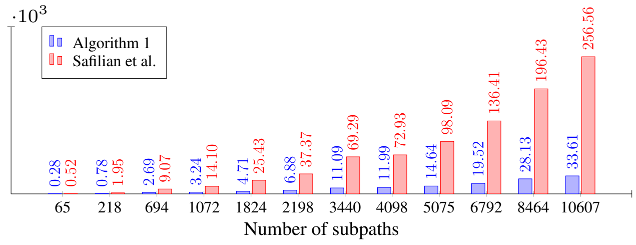

Figure 6: The length for SPP (excluding subpaths) (b).

approximation ratio worse in an additional step of the previous algorithm. We also observe that the tour lengths almost match the lower bounds given by minimum spanning trees, which means that our solutions are almost optimal.

Figure 7 reports the actual execution time. Both algorithms take $O(n^3)$ time, but ours is always faster than the previous one by around 30%. This is because our algorithm is simpler than the previous one. Note that an instance with up to 10,000 vertices can be solved within 20 minutes.

In summary, we observe that both the solution quality and the computation time of our algorithm outperform the previous one's.

5.2 Results for SGPP

Similarly to SPP, we use VLSI datasets for TSP. We define the set of subgroups on each instance by the following two ways: (a) make a group of neighboring vertices in a greedy way, (b) make a group in a random way. We set the sizes of subgroups to be all 3 or all 20. The length $c(u, v)$ of an inner edge (u, v) is set to be $X \cdot d(u, v)$, where $d(u, v)$ is the original distance between u and v and X is an integer drawn from $[2, 10]$ uniformly at random.

We implemented our $(1.5 + \alpha)$-approximation algorithm with $\alpha = 1$. In the algorithm, we solve TSPP exactly as

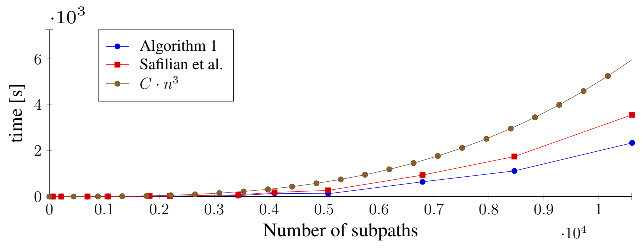

Figure 7: The execution time for SPP.

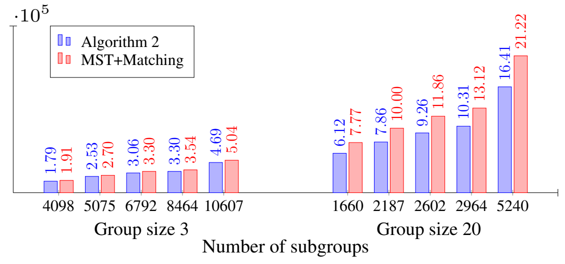

Figure 8: The length for SGPP (a).

follows. When the subgroup size is 3, we solve it by enumerating all the possible paths; when the subgroup size is 20, we use the dynamic programming algorithm [Bellman, 1962; Held and Karp, 1962] which runs in $O(k^2 2^k)$, where k is the subgroup size. Thus the returned solution is expected to be 2.5-approximation.

To compare with our algorithm, we implemented a naive heuristics based on Christofides' algorithm. We first find a tour in each subgroup S_i from a minimum spanning tree and a minimum-weight matching, and then find a tour after contracting each subgroup to one vertex in a similar way. Concatenating the obtained tours, we construct a feasible tour. This can be done in $O(n^3)$ time.

Figure 8 shows the solution quality of the two algorithms when subgroups are generated as (a). One can see that Algorithm 2 returns a better solution for all instances. The solution quality is improved by more than 10%. Interestingly, if the subgroup size is larger, the improvement becomes more significant. The results for (b) are similar, and so we omit them.

We also confirm that our algorithm runs in $O(n^3)$ time for a fixed subgroup size. Because we solve TSPP exactly, our algorithm takes much time when the subgroup size is 20; for example, Algorithm 2 took 4,877 seconds for an instance with 5,240 subgroups, while the heuristic 554 seconds.

6 Conclusion and Discussion

In this paper, we proposed a 1.5-approximation algorithm for SPP, which improves the previous result [Safilian *et al.*, 2016]. We also introduced SGPP as a generalization of SPP, and proposed a 3-approximation algorithm. Both of our results are obtained by utilizing the techniques developed for TSP and TSPP. Our numerical experiments showed the superiority of our approximation algorithms in terms of the solution quality and the computational time.

As we mentioned in the introduction, the metric SGPP is a generalization of the metric CTSP. There exists a 11/4-approximation algorithm for the metric CTSP [Guttmann-Beck *et al.*, 2000]. By incorporating the ideas used there, it seems possible to improve the approximation ratio for SGPP.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments, particularly for pointing out the connection to CTSP. This work is supported by JST ERATO Grant Number JPMJER1201, Japan.

References

[Arora, 1998] S. Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998.

[Bellman, 1962] R. Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, 1962.

[Chisman, 1975] J. A. Chisman. The clustered traveling salesman problem. *Comput. Oper. Res.*, 2(2):115–119, 1975.

[Christofides, 1976] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, USA, 1976.

[Dorigo and Gambardella, 1997] M. Dorigo and L. M. Gambardella. Ant colonies for the travelling salesman problem. *Biosystems*, 43(2):73–81, 1997.

[Ducomman *et al.*, 2016] S. Ducomman, H. Cambazard, and B. Penz. Alternative filtering for the weighted circuit constraint: Comparing lower bounds for the TSP and solving TSPTW. In *Proc. AAAI'16*, pages 3390–3396, 2016.

[Eiselt *et al.*, 1995] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part II: The rural postman problem. *Oper. Res.*, 43(3):399–414, 1995.

[Frederickson, 1979] G. N. Frederickson. Approximation algorithms for some postman problems. *J. ACM*, 26(3):538–554, 1979.

[Gabow, 1990] H. N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proc. SODA'90*, pages 434–443, 1990.

[Gottschalk and Vygen, 2016] C. Gottschalk and J. Vygen. Better s-t-tours by Gao trees. In *Proc. IPCO'16*, pages 126–137, 2016.

[Guttmann-Beck *et al.*, 2000] N. Guttmann-Beck, R. Hassin, S. Khuller, and B. Raghavachari. Approximation algorithms with bounded performance guarantees for the clustered traveling salesman problem. *Algorithmica*, 28(4):422–437, 2000.

[Gyorfi *et al.*, 2010] J. S. Gyorfi, D. R. Gamota, S. M. Mok, J. B. Szczech, M. Toloo, and J. Zhang. Evolutionary path planning with subpath constraints. *IEEE Trans. Electron. Packag. Manuf.*, 33(2):143–151, 2010.

[Held and Karp, 1962] M. Held and R. M. Karp. A dynamic programming approach to sequencing problems. *SIAM Rev. Soc. Ind. Appl. Math.*, 10(1):196–210, 1962.

[Hoogeveen, 1991] J. A. Hoogeveen. Analysis of Christofides' heuristic: Some paths are more difficult than cycles. *Oper. Res. Lett.*, 10(5):291–295, 1991.

[Jaillet and Porta, 2013] L. Jaillet and J. M. Porta. Path planning under kinematic constraints by rapidly exploring manifolds. *IEEE Trans. Robot.*, 29(1):105–117, 2013.

[Kapadia *et al.*, 2013] M. Kapadia, K. Ninomiya, A. Shoulson, F. Garcia, and N. Badler. Constraint-aware navigation in dynamic environments. In *Proc. MIG'13*, pages 89:111–89:120, 2013.

[Korte and Vygen, 2002] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 2002.

[Kumar *et al.*, 2013] T. K. S. Kumar, M. Cirillo, and S. Koenig. On the traveling salesman problem with simple temporal constraints. In *Proc. SARA'13*, 2013.

[Lin and Kernighan, 1973] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.*, 21(2):498–516, 1973.

[Lokin, 1979] F. C. J. Lokin. Procedures for travelling salesman problems with additional constraints. *Eur. J. Oper. Res.*, 3(2):135–141, 1979.

[Mitchell, 1999] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999.

[Mömke and Svensson, 2011] T. Mömke and O. Svensson. Approximating graphic TSP by matchings. In *Proc. FOCS'11*, pages 560–569, 2011.

[Mömke and Svensson, 2016] T. Mömke and O. Svensson. Removing and adding edges for the traveling salesman problem. *J. ACM*, 63(1):2:1–2:28, 2016.

[Nash *et al.*, 2009] A. Nash, S. Koenig, and M. Likhachev. Incremental Phi*: Incremental any-angle path planning on grids. In *Proc. IJCAI'09*, pages 1824–1830, 2009.

[Safilian *et al.*, 2016] M. Safilian, S. M. Hashemi, S. Eghbali, and A. Safilian. An approximation algorithm for the subpath planning problem. In *Proc. IJCAI'16*, pages 669–675, 2016.

[Sahni and Gonzalez, 1976] S. Sahni and T. Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, 1976.

[Surynek, 2015] P. Surynek. Reduced time-expansion graphs and goal decomposition for solving cooperative path finding sub-optimally. In *Proc. IJCAI'15*, pages 1916–1922, 2015.

[Tong-ying *et al.*, 2004] G. Tong-ying, Q. Dao-kui, and D. Zai-li. Research of path planning for polishing robot based on improved genetic algorithm. In *Proc. ROBIO'04*, pages 334–338, 2004.

[Tu *et al.*, 2010] D. Tu, S. Guo, H. Qin, W.-C. Oon, and A. Lim. The tree representation of feasible solutions for the TSP with pickup and delivery and LIFO loading. In *Proc. AAAI'10*, pages 191–196, 2010.

[Zhang and Looks, 2005] W. Zhang and M. Looks. A novel local search algorithm for the traveling salesman problem that exploits backbones. In *Proc. IJCAI'05*, pages 343–348, 2005.