

Improved Approximations for TSP with Simple Precedence Constraints^{*}

(Extended Abstract)

Hans-Joachim Böckenhauer¹, Ralf Klasing², Tobias Mömke¹, and
Monika Steinová¹

¹ Department of Computer Science, ETH Zurich, Switzerland,
{hjb,tobias.moemke,monika.steinova}@inf.ethz.ch

² CNRS - LaBRI - Université Bordeaux 1, 351 cours de la Libération, 33405 Talence
cedex, France, klasing@labri.fr

Abstract. In this paper, we consider variants of the traveling salesman problem with precedence constraints. We characterize hard input instances for Christofides' algorithm and Hoogeveen's algorithm by relating the two underlying problems, i. e., the traveling salesman problem and the problem of finding a minimum-weight Hamiltonian path between two prespecified vertices. We show that the sets of metric worst-case instances for both algorithms are disjoint in the following sense. There is an algorithm that, for any input instance, either finds a Hamiltonian tour that is significantly better than 1.5-approximative or a set of Hamiltonian paths between all pairs of endpoints, all of which are significantly better than $5/3$ -approximative.

In the second part of the paper, we give improved algorithms for the ordered TSP, i. e., the TSP, where the precedence constraints are such that a given subset of vertices has to be visited in some prescribed linear order. For the metric case, we present an algorithm that guarantees an approximation ratio of $2.5 - 2/k$, where k is the number of ordered vertices. For near-metric input instances satisfying a β -relaxed triangle inequality, we improve the best previously known ratio to $k\beta^{\log_2(3k-3)}$.

1 Introduction

Many practically relevant problems in operations research can be formalized by the means of the traveling salesman problem (TSP) or one of its many generalizations [7]. While the classical TSP asks for an arbitrary minimum-weight Hamiltonian tour in a complete edge-weighted graph, we will consider a class of TSP variants here, where the set of feasible tours is restricted by some precedence constraints, i. e., by demanding that certain vertices have to be visited before certain others. In principle, such precedence constraints can be expressed by an arbitrary partial ordering on the vertices, but we will restrict our attention to two simple special cases here: The first variant we are considering here

^{*} The research was partially funded by the ANR-projects "ALADDIN" and "IDEA", the INRIA project "CEPAGE", and by SNF grant 200021-109252.

is the well-known path TSP with given endpoints, i. e., the problem of finding a minimum-weight Hamiltonian path between two prespecified endpoints. The second variant is the so-called ordered TSP (k -OTSP) [3] where a linear order on a subset of k vertices is given as part of the input and every feasible solution has to contain these special vertices in the prescribed order.

While being one of the hardest problems with respect to approximability in its general formulation [10], the metric TSP is well-known to be approximable within a constant factor of 1.5 due to Christofides' algorithm [4]. Christofides' algorithm has been generalized by Hoogeveen [9] to work also for the problem of finding a Hamiltonian path with prespecified endpoints. This generalized algorithm achieves an approximation ratio of $5/3$. A different proof for the same result was given by Guttmann-Beck et al. [8]. Both of these upper bounds on the approximability have resisted all attempts of improvement for many years.

In this paper, we characterize hard input instances for both algorithms by relating these two problems. We show that the sets of worst-case instances for Christofides' algorithm and Hoogeveen's algorithm are disjoint in the following sense: We describe an algorithm that, for any metric input instance, either finds a Hamiltonian tour that is significantly better than 1.5-approximative or a set of Hamiltonian paths between all pairs of endpoints, all of which are significantly better than $5/3$ -approximative (see Figure 1). This result relates to the design of hybrid algorithms as proposed by Vassilevska et al. [11], as well as to the win/win strategy which is popular for designing parametrized algorithms [6]. Very recently, a similar approach of relating the approximability of different optimization problems has been investigated by Eppstein [5].

For the k -OTSP, we consider both the metric case and the near-metric case where the edge-weights satisfy a β -relaxed triangle inequality, i. e., where, for some $\beta > 1$, the edge-weight function c satisfies the condition $c(\{u, v\}) \leq \beta \cdot (c(\{u, w\}) + c(\{w, v\}))$ for any pairwise distinct vertices u, v , and w . In both cases, we improve the approximability results from [3], where a 2.5 approximation algorithm for the metric case and a $((k+1) \cdot \min\{4\beta^{2+\log_2(k-1)}, 1.5\beta^{3+\log_2(k-1)}, (\beta+1)\beta^{2+\log_2(k-1)}\})$ approximation algorithm for the near metric case are presented. Note that for k equal to one or two, any Hamiltonian tour respects the order of the special vertices. For k greater than two, our main results in this part of the paper are as follows: For the metric case, we present an algorithm that guarantees an approximation ratio of $2.5 - 2/k$, where k is the number of ordered vertices. For near-metric input instances satisfying a β -relaxed triangle inequality, we improve the approximation ratio to $k\beta^{\log_2(3k-3)}$. Due to space limitations, some proofs are omitted in this extended abstract.

2 Preliminaries

In a simple graph $G = (V, E)$, the edges are sets of two vertices. We use, however, a shortened notation. Instead of $\{u, v\}$, we simply write an unordered pair uv , where u and v are vertices. For a graph G , we refer to the sets of vertices and edges using $V(G)$ and $E(G)$, respectively. A *trail* from u to v is a sequence of

adjacent edges leading from u to v , where no edge may be used more than once. A trail is uniquely defined by a list of vertices $uw_1w_2\dots w_iv$, where consecutive vertices describe the edges of the trail. We say that $w_1\dots w_i$ are the *inner vertices*. The *length of a trail* is the number of its edges. A trail, where each vertex is used at most once, is a *path*. A closed trail, i.e., a trail that starts and ends with the same vertex, is a *circuit*. A circuit, where each inner vertex is visited only once, is a *cycle*. In a graph $G = (V, E)$, a *Hamiltonian path* from u to v is a path of length $|V| - 1$ from u to v and a *Hamiltonian tour* is a cycle of length $|V|$. In a tree T , $P_T(u, v)$ denotes the unique path between two vertices u and v . Let $[n]$ denote the set $\{1, 2, \dots, n\}$, where n is an integer. A trail of length l respects the order of a tuple (v_1, \dots, v_k) of vertices, if there is an injective mapping $f : [k] \rightarrow [l + 1]$ such that v_i is the $f(i)$ -th vertex of the trail and either $f(i) < f(j)$ for all $i < j$ or $f(i) > f(j)$ for all $i < j$. A circuit respects the order of a tuple, if there is a vertex v such that starting from v , the corresponding trail respects the order of the tuple.

We call a complete graph $G = (V, E)$ with cost function $c : E \rightarrow \mathbb{Q}^+$ *metric*, if the edge costs satisfy the triangle inequality $c(uv) \leq c(uw) + c(wv)$ for any pairwise distinct vertices $u, v, w \in V$.

The *cost* of a graph or of a trail is the sum of the costs of its edges. For simplicity, we write $c(X)$ for the cost of X , where X can be a graph or a trail.

Given a graph $G = (V, E)$ and a vertex $v \notin V$, we define $G + v$ as $(V \cup \{v\}, E \cup \{vw \mid w \in V\})$. Furthermore, given two vertices u and v in G , then we define $G + uv$ as $(V, E \cup uv)$. Note that adding vertices to a complete graph results in a complete graph. A vertex is *even* or *odd*, if its degree is even or odd, and a graph is even or odd, if all its vertices are even or odd, respectively.

The *metric traveling salesman problem*, Δ TSP, is the problem of finding a minimum-cost Hamiltonian tour in a complete metric graph. We also consider some variations of the Δ TSP. The *metric minimum-cost Hamiltonian path problem* in complete graphs, where the two end vertices are fixed, is called Δ HPP₂.

The *ordered metric traveling salesman problem*, k - Δ OTSP, is a generalization of the Δ TSP. As in the Δ TSP, we also search for a Hamiltonian tour in the given graph, but we require additionally that some special vertices appear in a predefined order within the tour. Formally, let $G = (V, E)$ be a complete metric graph with cost function $c : E \rightarrow \mathbb{Q}^+$, and let (s_1, s_2, \dots, s_k) be a k -tuple of vertices of V . Then the k - Δ OTSP is the problem of finding a minimum-cost Hamiltonian tour in G respecting the order of (s_1, s_2, \dots, s_k) .

We call the k - Δ OTSP with β -relaxed triangle inequality k - Δ_β OTSP. For both problems, k - Δ OTSP and k - Δ_β OTSP, we assume without loss of generality that k is greater than two.

3 A Win/Win Strategy for TSP and Hamiltonian Paths

Currently, the algorithm of Christofides [4] has the best proven approximation ratio for the Δ TSP, which is 1.5. A slight modification of that algorithm was shown by Hoogeveen [9] to be 5/3-approximative for the Δ HPP₂. In this section,

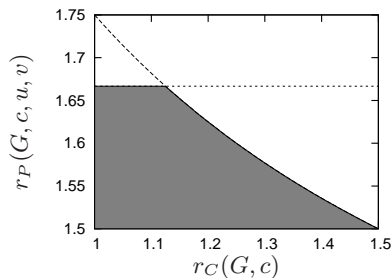


Fig. 1. Upper bound on the approximation ratio achieved by Algorithm 1. The horizontal line displays the approximation ratio $r_P(G, c, u, v) \leq 5/3$ proven in [9].

Algorithm 1 Path and Cycle

Input: A complete graph $G = (V, E)$, a metric cost function $c : E \rightarrow \mathbb{Q}^+$, and two vertices u and v .

- 1: Compute a minimum spanning tree T in G .
- 2: Compute a minimum perfect matching M_C on the odd vertices of T in G .
- 3: Compute a minimum perfect matching M_P on the odd vertices of the multigraph $T + uv$ in G .
- 4: Compute an Eulerian tour E_C in the multigraph $T \cup M_C$ and an Eulerian path E_P in the multigraph $T \cup M_P$.
- 5: Shorten E_C and E_P to a Hamiltonian tour H_C and a Hamiltonian path H_P respectively.

Output: H_C and H_P .

we show that the two mentioned problems are strongly related: for any given metric graph, either we can find a better approximation than 1.5 for Δ TSP or we can solve the Δ HPP₂ better than 5/3-approximately, whichever end-vertices we choose. For this purpose, we present Algorithm 1 which combines Christofides' and Hoogeveen's algorithm.

Considering Algorithm 1, let $r_C(G, c) := c(H_C)/c(\text{OPT}_C(G, c))$ be the approximation ratio for the computed Hamiltonian tour and let $r_P(G, c, u, v) := c(H_P)/c(\text{OPT}_P(G, c, u, v))$ be the approximation ratio for the computed Hamiltonian path for a given input G, c, u, v , where $\text{OPT}_C(G, c)$ and $\text{OPT}_P(G, c, u, v)$ are optimal solutions for the Δ TSP and the Δ HPP₂, respectively.

Theorem 1. *In Algorithm 1, the approximation ratio $r_P(G, c, u, v)$ is at most $1 + 1.5/(2r_C(G, c))$, independent of the choice of u and v . In particular, only pairs of approximation ratios from the shaded area in Figure 1 are possible.*

Proof. We distinguish two cases according to the cost of $\text{OPT}_C(G, c)$ in relation to the cost of $\text{OPT}_P(G, c, u, v)$.

Case I: Suppose that $c(\text{OPT}_C(G, c)) \leq c(\text{OPT}_P(G, c, u, v))$ holds.

Since both $\text{OPT}_C(G, c)$ and $\text{OPT}_P(G, c, u, v)$ contain a spanning tree, we con-

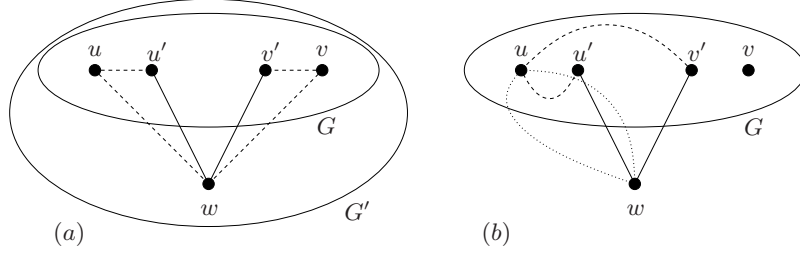


Fig. 2. The transformation removes the solid lines and inserts the dashed lines. In (a), the dashed lines belong to the computed Eulerian cycle in G' and in (b) they belong to the Eulerian cycle in G . The dotted lines in (b) are used for the construction, but they are not part of the solution.

clude that $c(T) \leq c(\text{OPT}_C(G, c)) \leq c(\text{OPT}_P(G, c, u, v))$ holds. Similar as in the analysis of Christofides' algorithm [4], we can also estimate the cost of M_P : given a set S containing an even number of vertices from V , we focus on the cycle $\text{OPT}_C(G, c)$, where the vertices from $(V \setminus S)$ are skipped. Due to the metricity, the formed cycle cannot be more expensive than $\text{OPT}_C(G, c)$. Furthermore, it contains two edge-disjoint perfect matchings of the vertices from S . Thus, the smaller one is at most half as expensive as $\text{OPT}_C(G, c)$. Since both M_C and M_P are minimal perfect matchings of an even number of vertices from V , we conclude that M_C as well as M_P cost at most $c(\text{OPT}_C(G, c))/2$. The remaining steps of the algorithm do not increase the costs of the solutions. Thus, we get a 1.5-approximative solution for both the path and the cycle.

Case II: Suppose that $c(\text{OPT}_C(G, c)) > c(\text{OPT}_P(G, c, u, v))$ holds.

Instead of directly using Algorithm 1, we first construct an auxiliary algorithm for which we will prove our claim. We will show afterwards that Algorithm 1 is always better than the auxiliary algorithm. Let $G' = (V', E')$ be the complete graph $G + w$, where w is a new vertex, and let $c' : E \rightarrow \mathbb{Q}^+$ be the cost function defined by $c'(e) = c(e)$ for all $e \in E$, $c'(wu) = c'(wv) = \sum_{e \in E} c(e)$, and $c'(wa) = \min\{c(wu) + c(ua), c(wv) + c(va)\}$ for any $a \in V - u - v$. Note that this way all edges wa are maximal with respect to the metricity (i. e., the resulting graph is metric and the metricity would be violated by any higher cost of wa).

First, we give an algorithm that transforms a given Hamiltonian tour \tilde{H} in G' either into another Hamiltonian tour in G' that costs at most $c'(\tilde{H})$ and contains wu and wv , or into a Hamiltonian tour in G that costs at most $c'(\tilde{H}) - c'(wu) - c'(wv)$. In \tilde{H} , there are two edges incident to w . Let wu' and wv' be these edges. Let without loss of generality $c(uu') \leq c(vv')$. The algorithm removes the edges wu' and wv' from \tilde{H} . If $c(vv') \leq c(uv')$, then the algorithm adds the path $u'uwvv'$ to \tilde{H} . Otherwise, if $c(vv') > c(uv')$, the algorithm adds the path $u'wv'$ to \tilde{H} , see Figure 2. In both cases, the transformation might result in nodes that are visited more than once. But due to the metricity, the resulting circuit can

be easily shortened either to a Hamiltonian tour in G' containing wu and wv or to a Hamiltonian tour in G .

Now we will show that the constructed tour is sufficiently cheap. In the algorithm, we assumed that $c(uu') \leq c(vu')$ holds. Thus, $c(wu') = c(wu) + c(uu')$. If also $c(vv') \leq c(uv')$ holds, then we analogously get $c(wv') = c(wv) + c(vv')$. Otherwise, if $c(vv') > c(uv')$, then $c(wv') = c(wu) + c(uv')$. In both cases, the shortening of the circuit does not increase its cost. Thus, the path $u'uwvv'$ or the path $u'uv'$ fulfils our cost requirements.

We conclude that there is an optimal Hamiltonian tour in G' that contains wu and wv ; the second option of the transformation, which is the Hamiltonian tour in G , would contradict the assumption that $c(\text{OPT}_P(G, c, u, v)) < c(\text{OPT}_C(G, c))$ in G , since $\text{OPT}_P(G, c, u, v)$ together with wu and wv would form a cheaper Hamiltonian tour.

We are now ready to present the main part of our algorithm. We construct a Hamiltonian tour H'_A in G' using Christofides' algorithm [4]. Then we transform the solution as described above. If it contains wu and wv , we remove these edges and output the remaining graph as H_P ; for H_C we apply Christofides' algorithm separately. If the transformation results in a Hamiltonian tour for G , we output that tour as H_C and construct H_P separately using Hoogeveen's algorithm [9].

Next, we show that the algorithm fulfils the requirements. Let T' be the minimum spanning tree and let M' be the minimum perfect matching constructed when applying Christofides' algorithm on G' . The tree T' costs at most $c(\text{OPT}_P(G, c, u, v)) + c'(wu)$, since adding the edge wu to $\text{OPT}_P(G, c, u, v)$ forms a spanning tree of G' . For an upper bound on the cost of M' , consider the Hamiltonian tour H' in G' formed by $\text{OPT}_P(G, c, u, v) + wu + wv$. Then, due to the metricity, $c(M') \leq c(\text{OPT}_P(G, c, u, v))/2 + (c(wu) + c(wv))/2$. Therefore, the cost of H'_A is at most $1.5c(\text{OPT}_P(G, c, u, v)) + 1.5c'(wu) + 0.5c'(wv) = 1.5c(\text{OPT}_P(G, c, u, v)) + c'(wu) + c'(wv)$. Therefore, in the first case of the transformation, removing wu and wv yields a Hamiltonian path H_P of cost at most $1.5c(\text{OPT}_P(G, c, u, v))$. Thus, both H_P and H_C are 1.5-approximative solutions. In the second case of the transformation, a similar argumentation as in the first case bounds $c(H_C)$ from above by $1.5c(\text{OPT}_P(G, c, u, v))$. Together with the definition of $r_C(G, c)$, we get

$$c(\text{OPT}_C) = c(H_C)/r_C(G, c) \leq 1.5c(\text{OPT}_P)/r_C(G, c).$$

Fact 1 *In G' , any minimum spanning tree can be decomposed into a minimum spanning tree in G and either the edge wu or the edge wv , since all other edges connecting w are more expensive and, due to the high cost for connecting w , the degree of w is one in any minimum spanning tree. Moreover, we can extend any minimum spanning tree in G to a minimum spanning tree in G' by simply adding the edge wu .*

Let S be the set of odd vertices in $T + uv$, where T is the minimum spanning tree within G that is formed by removing one edge from T' , which exists according to Fact 1. Thus, for any minimum perfect matching M of S in T , there is an Eulerian tour from u to v in $T \cup M$. The tour $\text{OPT}_C(G, c)$ contains

two edge-disjoint perfect matchings of S in T . Since $c(T) \leq c(\text{OPT}_P(G, c, u, v))$ holds, we can bound $c(H_P)$ now from above by

$$\begin{aligned} c(H_P) &\leq c(\text{OPT}_P(G, c, u, v)) + c(\text{OPT}_C(G, c)) / 2 \\ &\leq c(\text{OPT}_P(G, c, u, v)) + \frac{1.5c(\text{OPT}_P(G, c, u, v))}{2r_C(G, c)}. \end{aligned}$$

It remains to show that the solution computed by the algorithm that we used for the analysis costs at least as much as the solution of Algorithm 1. Due to Fact 1, we can assume without loss of generality that the minimum spanning tree computed by the transformation algorithm is $T + wu$. Thus, in both algorithms, we modify the degrees of the vertices of T , one time in order to obtain an even graph for the cycle and a second time in order to obtain a graph containing a trail from u to v . Due to the metricity, in both cases, without changing the tree we cannot do better than computing a minimum perfect matching on the nodes of wrong degree. Therefore, the result of Algorithm 1 cannot be worse than the result of the algorithm used in the proof. \square

Instead of $r_P(G, c, u, v)$, we can consider the parameter $c(M_P)/c(T)$ to distinguish different cases. This way, the analysis of the win/win strategy results in higher approximation ratios, but we know which of the two solutions has an improved approximation ratio.

4 Metric Ordered TSP

Our approximation algorithm for k - Δ OTSP is based on the following idea. We obtain a multigraph by combining a minimum spanning tree and a cycle formed by the ordered vertices (compare [3]). In the cycle, however, we skip the two most expensive edges. Afterwards, we obtain an Eulerian graph by adding a minimum perfect matching on the odd vertices to the multigraph. Within this Eulerian graph, we compute an Eulerian tour that respects the order of the input k -tuple. Finally, we shorten the Eulerian tour in order to obtain a Hamiltonian tour that respects the order of the k -tuple.

Theorem 2. *Algorithm 2 is a polynomial-time $(2.5 - 2/k)$ -approximation algorithm for k - Δ OTSP, where k is the number of ordered vertices.*

In the proof of Theorem 2, we show that, due to the degrees of vertices in the considered subgraphs, all paths and cycles constructed within the algorithm must exist and that we can shorten the constructed Eulerian tour without violating the order of t . Finally, we show that the avoided edges e_1 and e_2 are expensive enough to guarantee the claimed approximation ratio.

5 Near-Metric Ordered TSP

Until now, we focused only on graphs with metric cost function. In fact, our approaches inherently depended on the triangle inequality. Now, in this section, we

Algorithm 2 k - Δ OTSP

Input: A complete graph G , a metric cost function c , and a k -tuple $t = (s_1, s_2, \dots, s_k) \subseteq V^k$.

- 1: Compute a minimum spanning tree T in G .
- 2: $C := s_1 s_2 \dots s_k s_1$.
- 3: Let e_1 and e_2 be the two most expensive edges of C and let $C' := C - e_1 - e_2$.
- 4: Let P be the path connecting the vertices that are incident to e_1 in T .
- 5: Compute a minimum perfect matching M on the odd vertices in the multigraph $A := T \cup C'$.
- 6: Let P' be the path connecting the vertices that are incident to e_2 in $A \cup M \setminus (C' \cup P)$.
- 7: Starting from the circuit $C' \cup P \cup P'$, compute an Eulerian tour in $A \cup M$ that respects the order of t .
- 8: Shorten the Eulerian tour to a Hamiltonian tour that respects the order of t .

Output: The computed Hamiltonian tour.

present an algorithm that computes an approximative solution for k - Δ_β OTSP. Figure 3 shows a comparison of our result and the best previously known approximation ratios.

In the first step of Algorithm 3, we build a minimum spanning tree T and we arbitrarily choose one vertex r of T as its root. Then, starting with r , we color all vertices except the special vertices s_1, \dots, s_k , i. e., those vertices that have to occur in prescribed order in the solution, in a breadth-first-search manner by a coloring function $f(v) = (f(\text{nearest colored predecessor}) \bmod k) + 1$. From now on, we omit to write colorings modulo k , when it is clear from the context. The values are always in the set $[k]$.

The paths L_i in Algorithm 3 connecting the vertices s_i and s_{i+1} use the coloring for bounding the distance between consecutive vertices of L_i from above.

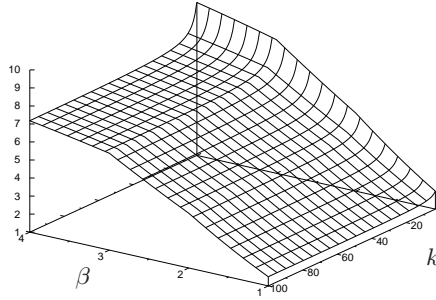


Fig. 3. The graph shows the quotient of the previously best known approximation ratio $(k + 1) \cdot \min\{4\beta^{2+\log_2(k-1)}, 1.5\beta^{3+\log_2(k-1)}, (\beta + 1)\beta^{2+\log_2(k-1)}\}$ and the ratio $k\beta^{\log_2 3(k-1)}$ achieved by Algorithm 3, i. e., the improvement achieved by the algorithm.

Algorithm 3 k - Δ_β OTSP

Input: A complete graph $G = (V, E)$ with edge weights $c : E \rightarrow \mathbb{Q}^+$ that satisfies the β -relaxed triangle inequality, and a tuple of vertices from V , (s_1, s_2, \dots, s_k) .

- 1: Build a minimum spanning tree T of graph G and choose a root r of T .
- 2: Define a coloring function $f : V \setminus \{s_1, \dots, s_k\} \rightarrow [k]$ as follows. Color r by color 1. Continue with coloring the vertices in breadth-first-search manner: For each vertex u , let v be the nearest ancestor in T that is not in $\{s_1, \dots, s_k\}$. Color vertex u by the color $(f(v) \bmod k) + 1$.
- 3: For $i = 1, 2, \dots, k$, let L_i be the path formed by $P_T(s_i, s_{i+1})$, picking only vertices with color i , starting with s_i and ending with s_{i+1} .
- 4: Create new paths L'_1, \dots, L'_k from L_1, \dots, L_k by including the vertices bypassed by all L_i s as shown in Figure 5 and discussed later.
- 5: Create a cycle C' by connecting paths L'_1, \dots, L'_k .
- 6: Let T_1, \dots, T_q with roots r_1, \dots, r_q be the maximal subtrees of T such that vertices in $T_i \setminus r_i$ are not included in C' and $r_i \in V(C')$. Let e_i^* be the cheapest edge incident to r_i in T_i . Use procedure HCT³ (refined) from [1] on each pair (T_i, e_i^*) for $1 \leq i \leq q$ to obtain Hamiltonian tours H_i of the vertices of T_i .
- 7: For $1 \leq i \leq q$, merge H_i with C' in r_i using e_i^* to construct a Hamiltonian tour H .

Output: The Hamiltonian tour H .

Lemma 1. *In step 3 of Algorithm 3, at most $3k - 4$ edges of T are bypassed between two consecutive vertices of color i while picking vertices for path L_i .*

If we connect the paths L_1, \dots, L_k now, we obtain a cycle C containing each vertex at most once and the special vertices respect the given order. But here, we have the problem that C might not contain all vertices of G . There are two types of vertices not connected to C :

1. vertices in the paths in T between two consecutive special vertices that are bypassed but not used (see the left part of Figure 4). Formally, we define the set of these vertices as $W = \{v \in V \mid \forall i (1 \leq i \leq k) v \notin V(L_i) \wedge \exists j (1 \leq j \leq k) v \in V(P_T(s_j, s_{j+1}))\}$.
2. vertices in subtrees of tree T where no special vertex is located and thus no path is passing this part of the tree (see the right part of Figure 4). Formally $Y = \{v \in V \mid \forall i (1 \leq i \leq k) v \notin V(P_T(s_i, s_{i+1}))\}$.

Now, we show how to merge all of vertices of W into the paths L_i , thereby constructing the new paths L'_i . We process the paths L_i in order, starting with L_1 . After creating the updated path L'_i , let W_i be the set of vertices of W that are still not connected to any of the paths, with $W_0 = W$.

Processing of the path L_i works as follows (see Figure 5). Let xy be an arbitrary edge in L_i , and let $P_T(x, y) = xv_1v_2 \dots v_z y$ be the corresponding path in the tree. Let p_{xy} be the maximal index such that for all $1 \leq j \leq p_{xy}$, $v_j \in W_{i-1}$. Similarly, let q_{xy} be the minimal index such that $\forall j, p_{xy} < q_{xy} \leq j \leq z$, $v_j \in W_{i-1}$. If p_{xy} and q_{xy} exist, we remove the edge xy from L_i and replace it by the path $xv_1v_2 \dots v_{p_{xy}}v_{q_{xy}} \dots v_z y$. In the case when p_{xy} or q_{xy} does not exist, $v_1 \dots v_{p_{xy}}$ or $v_{q_{xy}} \dots v_z$ is empty. Then we replace xy by $xv_{q_{xy}}, \dots, v_z y$

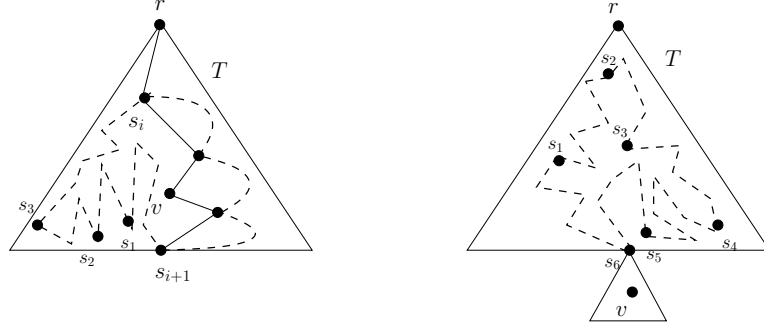


Fig. 4. The triangle denotes tree T , dashed lines denote paths L_1, \dots, L_k and the line between s_i and s_{i+1} is the path $P_T(s_i, s_{i+1})$. The two possible cases where vertex v is not connected to paths L_i are shown here. Either vertex v lies on some path $P_T(s_i, s_{i+1})$ but was not picked to some path L_j ($1 \leq j \leq k$) (left part) or vertex v is in a subtree where no path L_j is passing (right part).

or $xv_1 \dots v_{p_{xy}}y$, respectively. The new path L'_i is constructed by repeating this process for every edge in L_i . We set $W_i = W_{i-1} \setminus V(L'_i)$. Note that vertices from W_i that are located between p_{xy} and q_{xy} , if any, are added to some later path L'_h . This follows from the fact that, since $p_{xy} < q_{xy}$, there exists an edge (v_{d_1}, v_{d_2}) in L_h ($i < j \leq k$ and $p_{xy} < d_1 < q_{xy}$) that is processed later.

Using the described approach, we include vertices from W into the new paths L'_1, \dots, L'_k . Then we join these paths to the cycle C' . Cycle C' passes the special vertices in required order and each vertex in W is included there exactly once. Therefore, later on we deal with the remaining non-connected vertices (set Y).

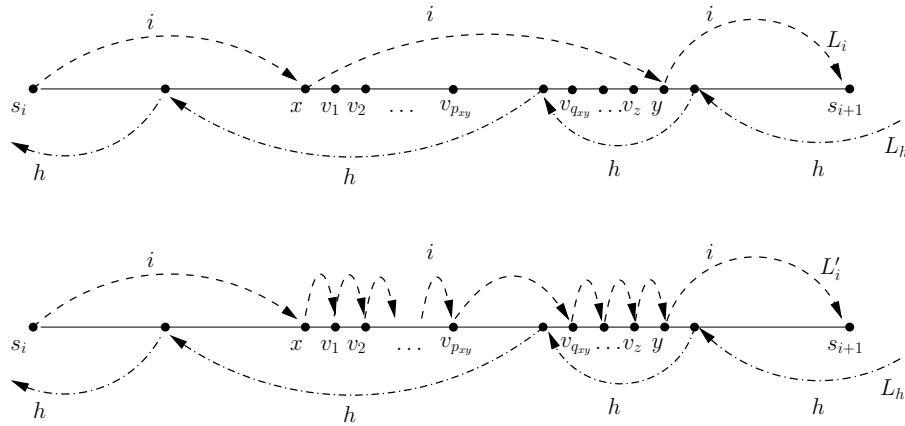


Fig. 5. Modifying path L_i to path L'_i .

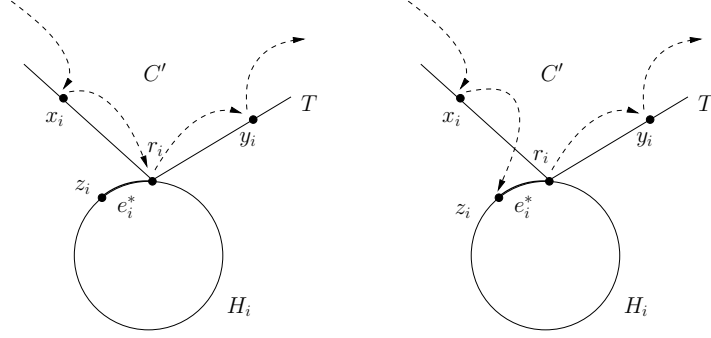


Fig. 6. Vertex r_i is the root of tree T_i , vertices x_i and y_i are the predecessor and successor of r_i in C' , respectively. Edge $e_i^* = (z_i, r_i)$ is the minimal edge incident to the root r_i from T_i . The Hamiltonian tour H_i is the tour built by procedure HCT^3 (refined) from [1] on the pair (T_i, e_i^*) . The merge of H_i to C' is done as follows: We reconnect the predecessor x_i of r_i to the vertex z_i . Thus, the Hamiltonian tour H continues from x_i to z_i , then it uses the Hamiltonian tour H_i , and finally, in r_i , it is connected back to the edges of C' .

The vertices in Y are organized in subtrees of T . We use this fact and we apply the algorithm HCT^3 (refined) from [1]. This algorithm is an approximation algorithm for the cheapest Hamiltonian tour in a complete graph with a cost function satisfying the relaxed triangle inequality. The algorithm builds a Hamiltonian tour on the vertices of a tree and bounds the cost of the created tour by $\beta + \beta^2$ times the cost of the tree. Moreover, the algorithm includes one particular edge of the tree to the created tour. This edge is then used to merge the created tour to our cycle C' . In the following paragraphs we will discuss the use of algorithm HCT^3 (refined) in more detail.

Let T_1, \dots, T_q be the maximal non-empty subtrees with roots r_1, \dots, r_q of T such that $V(T_i \setminus r_i) \cap V(C') = \emptyset$ (all vertices of the trees except their roots are not in C'). Let e_1^*, \dots, e_q^* be the cheapest edges of T_1, \dots, T_q that are incident with r_1, \dots, r_q (in [1] they are called *locally minimal*). We use the procedure HCT^3 (refined) with initial edges e_1^*, \dots, e_q^* to build the Hamiltonian tours H_1, \dots, H_q that contain all vertices from T_1, \dots, T_q — see step 6 of Algorithm 3.

The last part of the algorithm is the merge of H_1, \dots, H_q with C' . We use the property of the procedure HCT^3 (refined) that H_1, \dots, H_q contain the initial edges e_1^*, \dots, e_q^* respectively. These are edges of tree T and therefore we are able to merge the Hamiltonian tours to C' in order to create the Hamiltonian tour H without duplicating vertices and without significant increase of the total cost as it is shown in Figure 6.

The important property of the β -relaxed triangle inequality is that the direct connection between two vertices of a complete graph can be more expensive than a detour. On the other hand, using the β -relaxed triangle inequality, we can bound the cost of paths with bypassed vertices: according to [2], a single edge bypassing a path P of length l costs at most $\beta^{\lceil \log_2 l \rceil} \cdot c(P)$.

Lemma 2. *Let $G = (V, E)$ be a graph with cost function c that satisfies the β -relaxed triangle inequality. Let $P = v_0 v_1 \dots v_l$ be a path in G . For $0 = a_0 < a_1 < \dots < a_k = l$, where $1 \leq k \leq l$ holds, let $m := \max_{0 \leq i < k} \{a_{i+1} - a_i\}$. Then $\sum_{i=1}^{k-1} c(v_{a_i}, v_{a_{i+1}}) \leq \beta^{\log_2 m} \cdot c(P)$.*

Now we are ready to analyze the costs of the paths L'_i and the cost of covering the remaining vertices that are not in any path L'_i .

Lemma 3. *For $1 \leq i \leq k$, the cost of the path L'_i is bounded from above by $c(L'_i) \leq \beta^{\log_2(3^{k-4})} \cdot c(P_T(s_i, s_{i+1}))$.*

Lemma 4. *For $1 \leq i \leq q$, the cost of the Hamiltonian tour H_i is bounded from above by $c(H_i) \leq (\beta^2 + \beta) \cdot c(T_i)$.*

This implies the main result.

Theorem 3. *Algorithm 3 computes a $(k\beta^{\log_2 3^{k-1}})$ -approximative solution for k - Δ_β OTSP in polynomial time.*

References

1. Andreae, T.: On the traveling salesman problem restricted to inputs satisfying a relaxed triangle inequality. *Networks* 38(2), 59–67 (2001)
2. Bandelt, H.J., Crama, Y., Spieksma, F.C.R.: Approximation algorithms for multi-dimensional assignment problems with decomposable costs. *Discrete Appl. Math.* 49(1-3), 25–50 (1994)
3. Böckenhauer, H.-J., Hromkovič, J., Kneis, J., Kupke, J.: On the approximation hardness of some generalizations of TSP (extended abstract). In: Arge, L., Freivalds, R.V. (eds.) *SWAT 2006*. LNCS, vol. 4059, pp. 184–195. Springer, Berlin (2006)
4. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Tech. Rep. 388, Graduate School of Industrial Administration, Carnegie-Mellon University (1976)
5. Eppstein, D.: Paired approximation problems and incompatible inapproximabilities. In: Charikar, M. (ed.) *Proc. of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*. pp. 1076–1086. SIAM, New York (2010)
6. Fellows, M.R.: Blow-ups, win/win's, and crown rules: Some new directions in FPT. In: Bodlaender, H.L. (ed.) *WG 2003*. LNCS, vol. 2880, pp. 1–12. Springer, Berlin (2003)
7. Gutin, G., Punnen, A.P. (eds.): *The Traveling Salesman Problem and Its Variations*. Combinatorial Optimization, Springer, New York (2007)
8. Guttmann-Beck, N., Hassin, R., Khuller, S., Raghavachari, B.: Approximation algorithms with bounded performance guarantees for the clustered traveling salesman problem. *Algorithmica* 28(4), 422–437 (2000)
9. Hoogeveen, J.A.: Analysis of Christofides' heuristic: some paths are more difficult than cycles. *Oper. Res. Lett.* 10(5), 291–295 (1991)
10. Sahni, S., Gonzalez, T.F.: P-complete approximation problems. *J. ACM* 23(3), 555–565 (1976)
11. Vassilevska, V., Williams, R., Woo, S.L.M.: Confronting hardness using a hybrid approach. In: *Proc. of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*. pp. 1–10. SIAM, New York (2006)