Improved approximations for TSP with simple precedence constraints

Author(s):
Böckenhauer, Hans-Joachim; Mömke, Tobias; Steinová, Monika

Publication Date:
2013-07

Originally published in:

Journal of Discrete Algorithms 21, http://doi.org/10.1016/j.jda.2013.04.002 →

Improved Approximations for TSP with Simple Precedence Constraints[☆]

Hans-Joachim Böckenhauer[a], Tobias Mömke[b,1], Monika Steinová[a]

[a] *Department of Computer Science, ETH Zurich, Switzerland*
[b] *Department of Computer Science, Saarland University, Saarbrücken, Germany*

Abstract

In this paper, we consider the ordered TSP, a variant of the traveling salesman problem with precedence constraints, where the precedence constraints are such that a given subset of vertices has to be visited in some prescribed linear order. We give improved algorithms for the ordered TSP: For the metric case, we present a polynomial-time 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 ratio to $k\beta^{\log_2(\lfloor 3k/2 \rfloor + 1)}$.

Keywords: ordered TSP, traveling salesman problem, approximation, relaxed triangle inequality

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 [5]. 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 a simple special case here: 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.

[☆] The research was partially funded by SNF grant 200021-132510/1.

Email addresses: hjb@inf.ethz.ch (Hans-Joachim Böckenhauer), moemke@cs.uni-saarland.de (Tobias Mömke), monika.steinova@inf.ethz.ch (Monika Steinová)

[1] This work was done during the stay of this author at ETH.

While the TSP is one of the hardest problems with respect to approximability in its general formulation [7], its restriction to metric instances is well-known to be approximable within a constant factor of 1.5 due to Christofides' algorithm [4].

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 previously known 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, two or three, any Hamiltonian tour respects the order of the special vertices. For larger k, our main results 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 $k\beta^{\log_2(\lfloor 3k/2 \rfloor+1)}$.

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. The set of *neighbors* of vertex u is denoted as $N(u) = \{v \in V \mid uv \in E\}$. 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 $|V|$. 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 $f : [k] \to [l + 1]$ 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. For a more detailed introduction to graph theory, see [8].

We call a complete graph $G = (V, E)$ with cost function $c : E \to \mathbb{Q}^+$ *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.

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*, ΔTSP, is the problem of finding a minimum-cost Hamiltonian tour in a complete metric graph. We also consider some variations of the ΔTSP.

The *ordered metric traveling salesman problem*, k-ΔOTSP, is a generalization of the ΔTSP. As in the ΔTSP, we also search for a Hamiltonian tour in the given graph, but we additionally require 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 $c : E \to \mathbb{Q}^+$, and let $(s_1, s_2, \ldots, s_k)$ be a k-tuple of vertices of V. Then the k-ΔOTSP is the problem of finding a minimum-cost Hamiltonian tour in G respecting the order of $(s_1, s_2, \ldots, s_k)$. We often refer to $s_1, \ldots, s_k$ as the *special vertices* of G.

We call the k-ΔOTSP with β-relaxed triangle inequality k-Δ_βOTSP. For both problems, k-ΔOTSP and k-Δ_βOTSP, we assume without loss of generality that $k > 2$.

2.1. Algorithm of Andreae

In Section 4, we are using the $(\beta^2+\beta)$-approximation algorithm of Andreae [1] for computing a Hamiltonian cycle which contains all vertices of a given tree as a subroutine of our algorithm (step 12 of Algorithm 3). As we strongly depend on this algorithm, we give a brief overview of it. All the details regarding the proof of the approximation ratio can be found in [1]. In the sequel, we denote this algorithm as procedure HCT^3 (refined).

Before presenting the actual algorithm, we need to mention a simple definition.

Definition 1. Let G be a graph with edge cost function $c : E \to \mathbb{Q}^+$. An edge $e^* \in E(G)$ is called a *locally minimal* edge of (G, c) if there exists one endpoint x of e^* such that the cost of e^* does not exceed the cost of all other edges incident to x, i. e., $c(e^*) \leq c(xy)$, for all $y \in N(x)$.

The algorithm is presented as Algorithm 1. Its general idea is to perform the computation recursively – first a suitable edge of a tree is selected (it is the locally minimal edge). The edge is removed from the tree which separates the tree into two components (that are smaller trees). The computation is performed recursively in both smaller trees and then the resulting two smaller Hamiltonian cycles are merged via the selected edge into a larger cycle which is again Hamiltonian. The local minimality of the edge ensures that the connection of the two cycles into a single one is not too expensive and thus the approximation ratio is well bounded.

We would like to point out that relating the costs of the tree and the Hamiltonian cycle are crucial for our algorithm. The property pays off in the approximation factor analysis (Theorem 3), where it allows us to reasonably estimate the cost of our partial solution by the cost of the underlying subtree.

Theorem 1 (Andreae, [1]). *Algorithm 1 applied on a tree T computes a near-metric* ΔTSP *solution H on vertices $V(T)$ in time $\mathcal{O}(|V(T)|^2)$ and with the cost*

$$c(H) \leq (\beta^2 + \beta)c(T).$$

Algorithm 1 (HCT^3 (refined) by Andreae, [1])

Input: A complete undirected graph $G = (V, E)$ with edge cost function $c : E \to \mathbb{Q}^+$ satisfying the β-relaxed triangle inequality ($\beta > 1$), a tree T with $|V(T)| \geq 3$ and a locally minimal edge $e^* = (a_1, a_2)$ of T.

For $i \in \{1, 2\}$, let T_i be the component of $(V(T), E(T) \setminus \{e^*\})$ containing a_i.
for $i = 1$ to 2 **do**
    if $|V(T_i)| \geq 2$ **then**
        pick $a'_i \in V(T_i)$ such that (a_i, a'_i) is a locally minimal edge in T_i.
        let $e^*_i \coloneqq (a_i, a'_i)$.
    else
        let $a'_i \coloneqq a_i$.
    end if
    if $|V(T_i)| \geq 3$ **then**
        recursively compute a Hamiltonian cycle H_i of $V(T_i)$ using e^*_i.
        let $P_i \coloneqq H_i \setminus \{e^*_i\}$.
    else
        let $P_i \coloneqq T_i$.
    end if
end for
Merge paths P_1, P_2 and edges e^* and (a'_1, a'_2) into a cycle H.

Output: The Hamiltonian cycle H containing e^*.

Hence, Algorithm 1 is a $(\beta^2 + \beta)$-approximation algorithm for near-metric ΔTSP *with time complexity of $\mathcal{O}(|V(T)|^2)$.*

Sketch of the proof. It is easy to argue that the algorithm always produces a Hamiltonian cycle on $V(T)$. The proof can be done by induction on the size of the tree. The base case for $|V(T)| = 3$ is trivial. In the inductive step, where the two smaller Hamiltonian cycles are produced, the edge (a_i, a'_i) is removed from each of them and the remaining two paths are merged on their endpoints by $e^* = (a_1, a_2)$ and (a'_1, a'_2). Obviously, the produced graph is a cycle, no vertex is present here more than once and no vertex is forgotten to be added.

The proof of the approximation ratio of the algorithm is rather complex. Hence, we point the reader to the original paper [1] for details.

The time complexity of the algorithm is quadratic in the number of vertices as, by each division of the tree, we obtain two trees of size smaller by at least one vertex. Each call of the procedure takes $\mathcal{O}(|V(T)|)$ time and the recursion is terminated after $\mathcal{O}(|V(T)|)$ calls. □

3. Metric Ordered TSP

Our approximation algorithm for k-ΔOTSP 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

Algorithm 2 (k-ΔOTSP)

Input: A complete graph G, a metric cost function c, and a k-tuple $t = (s_1, s_2, \ldots, s_k) \in V^k$, for $k \geq 4$.

1: Compute a minimum spanning tree T in G.
2: $C := s_1 s_2 \ldots s_k s_1$.
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 in T connecting the vertices that are incident to e_1.
5: Compute a minimum perfect matching M on the odd vertices in the multigraph $A := T \cup C'$.
6: Let P' be the path in $A \cup M \setminus (C' \cup P)$ connecting the vertices that are incident to e_2.
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 respecting the order of t.

Output: The computed Hamiltonian tour.

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. Figure 1 illustrates the structure of our algorithm.

Theorem 2. *Algorithm 2 is a $\mathcal{O}(|E| \cdot \sqrt{|V|})$-time $(2.5 - 2/k)$-approximation algorithm for k-ΔOTSP, where $k \geq 4$ is the number of special vertices.*

Proof. We show that, due to the degrees of vertices in the considered subgraphs, all paths and cycles constructed within the Algorithm 2 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.

We will refer to the structures created in Algorithm 2. Let us consider the trail formed by $C' \cup P$, where P is used to connect the vertices adjacent to the removed edge e_1 (after line 4). Note that including e_2 into that trail would yield a circuit that respects the order of t. Instead of using e_2, however, we will use another trail. To this end, we add the minimum perfect matching M to A; this results in an Eulerian multigraph.

Let B be the graph $A \cup M$ without the edges of $C' \cup P$. Since the only odd vertices of $C' \cup P$ are the two that are incident to e_2, and since every vertex in $A \cup M$ has even degree, the only odd vertices of B are the vertices incident to e_2. These two vertices must be in the same component of B, because each component of a graph has an even number of odd vertices. Thus, there exists a trail P' connecting the vertices incident to e_2 in $A \cup M$ that is edge-disjoint to $C' \cup P$. Therefore, by closing the previously constructed trail using P', we

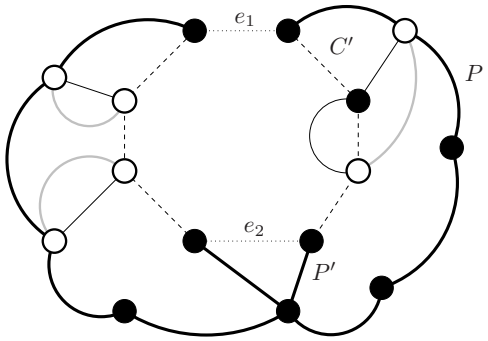

Figure 1: Some structures used in Algorithm 2. The dashed lines form C', the gray lines M, and the bold lines P and P'. The dotted lines are the removed edges e_1 and e_2. All solid black lines together form the spanning tree T. The non-filled vertices are the odd vertices in $T \cup C'$.

obtain a circuit in $A \cup M$ that respects the order of t (line 7).

Starting from that circuit, it is easy to construct an Eulerian tour in $A \cup M$ that respects the order of t: each component in $A \cup M$ without the edges of the constructed circuit is even and — since $A \cup M$ is connected — contains at least one vertex from the circuit. Thus, each component has an Eulerian tour that can be inserted into the constructed circuit.

Finally, we have to shorten the Eulerian tour to a Hamiltonian tour. We start from the vertex s_1. From there, we follow the Eulerian tour, but we use shortcuts to avoid vertices that have been visited already and vertices from t that are visited too soon. Skipping the vertices from t is justified, since we are sure that the Eulerian tour will eventually enter them again later on.

Now, let us analyze the approximation ratio of the algorithm. Due to the triangle inequality, the computed Hamiltonian tour is not more expensive than the underlying Eulerian tour, which contains each edge of $A \cup M$ exactly once. Therefore, its cost is at most $c(A) + c(M) = c(T) + c(C') + c(M)$. Let OPT be an optimal solution for k-ΔOTSP in G for the tuple t. Since OPT (as every TSP tour in G) contains a spanning tree, the cost of T is at most $c(\mathrm{OPT})$. Similarly, if we knew OPT, we could obtain C from it by skipping all vertices that are not contained in t. Therefore, due to the metricity, the cost of C is at most $c(\mathrm{OPT})$, too. Note that C has exactly k edges. Thus, $c(e_1) + c(e_2) \geq 2 \cdot c(C)/k$, and therefore the cost of C' is at most $(1 - 2/k) \cdot c(\mathrm{OPT})$. We bound the cost of the matching M similarly as in the analysis of Christofides' algorithm [4]: by skipping all vertices of OPT that are not in M, we obtain a cycle that is composed of the vertices of M and that costs at most $c(\mathrm{OPT})$. The cycle, however, contains two edge-disjoint perfect matchings. Therefore, the cost of M is at most $c(\mathrm{OPT})/2$. Overall, we can bound the cost of the computed Hamiltonian tour from above by $(2.5 - 2/k) \cdot c(\mathrm{OPT})$.

The computation of the minimum spanning tree can be done in time $\mathcal{O}(|E| \cdot \log |V|)$. The steps 2 and 3 can be performed in a linear time. The graph A has

$\mathcal{O}(|V|)$ vertices and edges. Hence due to algorithm of Micali and Vazirani [6], which computes a minimum perfect matching on a graph with n vertices and m edges in time $\mathcal{O}(m \cdot \sqrt{n})$, we can find matching M from step 5 in time $\mathcal{O}(|E| \cdot \sqrt{|V|})$. The computation of the Eulerian tour and its shortening can also be performed in a linear time. Hence, overall, the bottleneck of our computation is the computation of a minimum perfect matching which also determines the overall time complexity. □

4. Near-Metric Ordered TSP

Until now, we focused on graphs with metric cost function. In fact, our approaches inherently depended on the triangle inequality. In this section, we present an algorithm that computes an approximative solution for k-Δ_βOTSP. Figure 2 shows a comparison of our result and the best previously known approximation ratios.

Throughout this section, we will discuss the parts of Algorithm 3 in detail. We divide its analysis into several logical parts.

In the first step of Algorithm 3, we build a minimum spanning tree T, which has cost of at most the cost of an optimal solution. To create a Hamiltonian tour out of T, one of the main challenges is to use only edges that correspond to "short" paths in T. This is necessary because direct edges can be considerably more expensive than paths in graphs with β-relaxed triangle inequality. To this end, the first part of the algorithm (until line 9) creates a collection of paths that form a cycle with the special vertices ordered correctly, where no edge replaces a path of more than $\lfloor 3k/2 \rfloor$ edges in T.

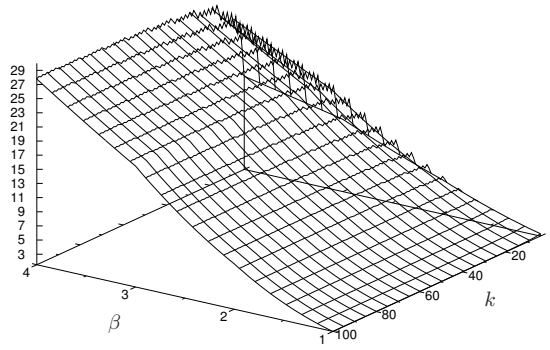

Figure 2: The graph shows the quotient of the previously best known approximation ratio $(k{+}1){\cdot}\min\{4\beta^{2+\log_2(k-1)}, 1.5\beta^{3+\log_2(k-1)}, (\beta{+}1)\beta^{2+\log_2(k-1)}\}$ and the ratio $k\beta^{\log_2(\lfloor 3k/2 \rfloor+1)}$ achieved by Algorithm 3, i. e., the improvement achieved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

Algorithm 3 (k-Δ_βOTSP)

Input: A complete graph $G = (V, E)$ with edge weights $c : E \to \mathbb{Q}^+$ satisfying the β-relaxed triangle inequality, and a tuple $(s_1, s_2, \ldots, s_k)$ of vertices from V, where $k \geq 4$.

1: Build a minimum spanning tree T of G and choose a root r of T.
2: For each pair of consecutive special vertices s_i, s_{i+1}, let P_i be the unique path in T connecting these vertices. Let P_k be the path connecting s_k and s_1.
3: Order the paths by increasing length. Rename the paths and the special vertices so that the index represents the order of the path (break ties arbitrarily). The renaming of the special vertices is done in such a way that path P_i starts at s_i.
4: Color each special vertex by its index.
5: For each i such that the length of P_i is at most $\lfloor 3k/2 \rfloor$, connect the ends of P_i directly forming a path L_i of length one. Let d be the highest index of such a path (or zero if there is none).
6: **for** $i = d + 1$ to k **do**
7: Let P_i be a path from s_i to s_j. Create L_i as follows. Starting from s_i, skip all colored vertices and $k - i$ uncolored vertices. Continue forming this type of edges until s_j is reached (the last edge might skip fewer uncolored vertices).
8: Color all vertices of L_i, except s_j, by i.
9: **end for**
10: Create new paths $L'_1, \ldots, L'_k$ from $L_1, \ldots, L_k$ by including the vertices bypassed by all L_is as shown in Figure 4 and discussed later.
11: Create a cycle C' by connecting the paths $L'_1, \ldots, L'_k$.
12: Let $T_1, \ldots, T_q$ with roots $r_1, \ldots, r_q$ be the maximal subtrees of T such that vertices in $T_i \backslash r_i$ 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 HCT^3 (refined) (see Section 2.1) 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.
13: 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 Algorithm 3, any edge in any path L_i corresponds to a path of T of length at most $\lfloor 3k/2 \rfloor$.*

PROOF. For the correctness, note that we merely need a path L_i for each P_i that connects the end vertices of P_i. The renaming does not change the pairs of end vertices. Therefore, C connects all special vertices in the correct order.

A jump of length i in G is an edge that directly connects two vertices that correspond to a path of length i in T. Now we analyze the length of the longest jump within the construction. Since line 5 of the algorithm is designed to construct only jumps of length at most $\lfloor 3k/2 \rfloor$, we only have to analyze the for-loop.

In the ith iteration of the loop, observe that, for each color $l < i$, there can be at most one vertex of color l that is skipped within one jump: If $l \leq d$, then there is only one vertex colored by l in G. Otherwise, let l' be the largest color smaller than i that was skipped. But between two vertices colored by l', there are $k - l' > k - i$ uncolored vertices. Thus the jump can contain at most one vertex colored by l'. By induction, it is not hard to see that there is also no color smaller than l' of which more than one vertex is skipped in the ith iteration.

Note that each time the last jump leads to a special vertex and thus to a different color. Therefore, the possibly smaller number of skipped uncolored vertices in the last jump does not harm the analysis.

Hence, each edge skips at most $k - i$ vertices colored by a number smaller than i, $i - 1$ uncolored vertices, and a number h of special vertices that have a higher number than i. Therefore, the maximal jump cannot contain more vertices than $k - i + (i - 1) + h = k - 1 + h$.

It remains to show that $h \leq \lfloor k/2 \rfloor$. Suppose, to the contrary, that $h > \lfloor k/2 \rfloor$. Then, within the at most $k - 1 + h$ skipped vertices there is a subpath of at most $\lfloor 3k/2 \rfloor$ vertices such that more than $\lfloor k/2 \rfloor$ of them have a color higher than i. (This is the case, since $\lfloor 3k/2 \rfloor - (k - 1) = \lfloor k/2 \rfloor + 1$.) But then, by the pigeonhole principle, at least two of them are consecutive special vertices. This, however, is a contradiction since consecutive vertices with a distance of at most $\lfloor 3k/2 \rfloor$ were already covered before the for-loop and thus have a lower index than i. □

If we now connect the paths $L_1, \ldots, L_k$, we obtain a cycle C containing each vertex at most once, and the special vertices respect the given order in C. There are two types of vertices not contained in 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 3). Formally, we define the set of these vertices as $W = \{v \in V \mid \forall i \ (1 \leq i \leq k) \ v \notin V(L_i) \wedge \exists j \ (1 \leq j \leq k) \ v \in V(P_j)\}$.
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 3). Formally, $Y = \{v \in V \mid \forall i \ (1 \leq i \leq k) \ v \notin V(P_i)\}$.

Now, we show how to merge all vertices of W into the paths L_i, thereby constructing the new paths L'_i. We process the paths L_i, starting with L_1. After

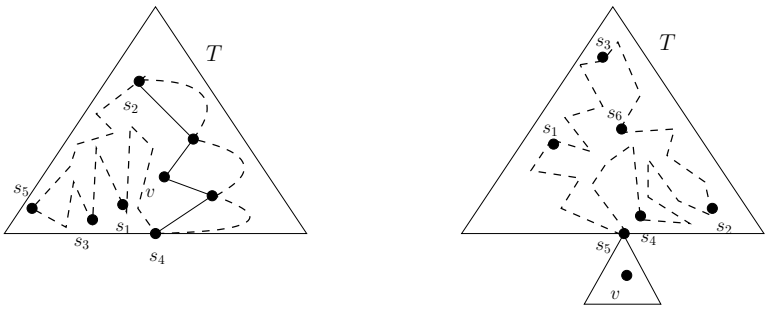

Figure 3: The triangle denotes the tree T, dashed lines denote paths $L_1, \ldots, L_k$ and the line between s_i and s_{i+1} is the path P_i (for simplicity, we do not consider the renaming done in line 3 of Algorithm k-Δ_βOTSP). The two possible cases where vertex v is not connected to paths L_i are shown here. Either vertex v lies on some path P_i 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).

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 4). Let xy be an arbitrary edge in L_i, and let $xv_1v_2 \ldots v_zy$ be the corresponding unique path in the tree T. 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 $xv_1v_2 \ldots v_{p_{xy}}v_{q_{xy}} \ldots v_zy$. In the case when p_{xy} or q_{xy} does not exist, $v_1 \ldots v_{p_{xy}}$ or $v_{q_{xy}} \ldots v_z$ is empty. Then we replace xy by $xv_{q_{xy}}, \ldots, v_zy$ or $xv_1 \ldots v_{p_{xy}}y$, 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

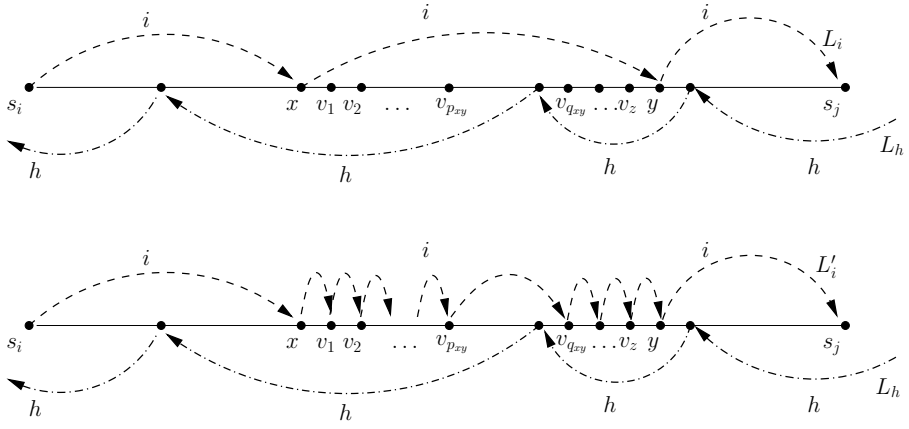

Figure 4: Modifying path L_i to path L'_i.

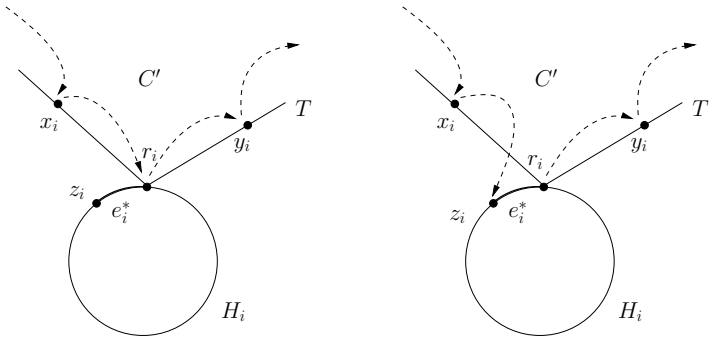

Figure 5: Vertex r_i is the root of tree T_i, vertices x_i and y_i are the predecessor and successor of r_i in C', respectively. Edge $e_i^* = (z_i, r_i)$ is the minimal edge incident to the root r_i from T_i. The Hamiltonian tour H_i is the tour built by procedure HCT^3 (refined) (see Section 2.1 for details) 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'.

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 all vertices from W into the new paths $L'_1, \ldots, L'_k$. Then we join these paths to the cycle C'. Cycle C' passes the special vertices in the 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).

The vertices in Y are organized in subtrees of T. We use this fact and we apply the algorithm HCT^3 (refined) (see Section 2.1).

Let $T_1, \ldots, T_q$ be the maximal non-empty subtrees with roots $r_1, \ldots, r_q$ 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 $e_1^*, \ldots, e_q^*$ be the locally minimal edges of $T_1, \ldots, T_q$ that are incident to $r_1, \ldots, r_q$. We use the procedure HCT^3 (refined) with initial edges $e_1^*, \ldots, e_q^*$ to build the Hamiltonian tours $H_1, \ldots, H_q$ that contain all vertices from $T_1, \ldots, T_q$ – see step 12 of Algorithm 3.

The last part of the algorithm is the merge of $H_1, \ldots, H_q$ with C'. We use the property of the procedure HCT^3 (refined) that $H_1, \ldots, H_q$ contain the initial edges $e_1^*, \ldots, e_q^*$, 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 5.

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 (Bandelt et al. [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 $0 = a_0 < a_1 < \cdots < a_k = l$, 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*

$$c(L'_i) \leq \beta^{\log_2 \lfloor 3k/2 \rfloor} \cdot c(P_i).$$

Proof. According to Lemma 1, the consecutive vertices of L_i are at most $\lfloor 3k/2 \rfloor$ vertices apart in the corresponding unique path in T. Including the additional vertices when creating L'_i from L_i does not increase the distance in P_i between two consecutive vertices. Thus, also in L'_i, the distance of consecutive vertices is at most $\lfloor 3k/2 \rfloor$. Therefore, by Lemma 2, the statement of this lemma is proved. □

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

Proof. Let T_i be a subtree of T from step 12 of Algorithm 3, let r_i be the root of T_i, and let e_i^* be the edge of minimal cost that is incident to r_i in T_i. Note that $r_i \in C'$ and $(V(T_i) \setminus r_i) \cap V(C') = \emptyset$. Since e_i^* is a minimal edge of T_i that is incident to the root r_i, it is locally minimal for T_i. Therefore, we can apply the procedure HCT^3 (refined) (see Section 2.1) to the tree T_i with the initial edge e_i^*.

Note that the procedure HCT^3 (refined) requires that the size of the input tree satisfies $|V(T_i)| \geq 3$. In the case $|V(T_i)| = 2$, our claim follows immediately. Otherwise, the output of the procedure HCT^3 (refined) is a Hamiltonian tour H_i that contains e_i^*. According to Theorem 1, $c(H_i) \leq (\beta^2 + \beta) c(T_i)$. □

This implies the main result of this section.

Theorem 3. *Algorithm 3 computes a $(k\beta^{\log_2(\lfloor 3k/2 \rfloor + 1)})$-approximative solution for k-Δ_βOTSP in quadratic time, for any $k \geq 4$.*

Proof. It is easy to see that Algorithm 3 computes a Hamiltonian tour where the special vertices are in the required order. Therefore, from now on we will estimate the approximation ratio and the running time. Since each vertex of C' is the root of at most one tree $T_1, \dots, T_q$, it can be bypassed at most once while connecting the Hamiltonian tours $H_1, \dots, H_q$ to C'. Since edge e_i^* $(1 \leq i \leq q)$ is an edge of T_i, it is an edge of T and therefore, applying Lemma 3, we can estimate the cost factor of reconnection of $H_1, \dots, H_q$ by $\gamma = \beta^{\log_2(\lfloor 3k/2 \rfloor + 1)}$.

Let us divide the edges of T into four disjoint groups:

$$\begin{aligned} S_1 &= \{e_i^* \in E \mid e_i^* \text{ is locally minimal, } e_i^* \in T_i,\ |T_i| = 2, 1 \le i \le q\} \\ S_2 &= \{e_i^* \in E \mid e_i^* \text{ is locally minimal, } e_i^* \in T_i.\ |T_i| > 2, 1 \le i \le q\} \\ S_3 &= \{e \in E \mid e \in T_i \setminus e_i^*,\ 1 \le i \le q\} \\ S_4 &= E(T) \setminus (S_1 \cup S_2 \cup S_3) \end{aligned}$$

Now we will prove that, for the factors $f_1 = \gamma + 1, f_2 = \beta^2 + \beta + \gamma, f_3 = \beta^2 + \beta$, and $f_4 = k\gamma$, the inequality $c(H) \le f_1 c(S_1) + f_2 c(S_2) + f_3 c(S_3) + f_4 c(S_4)$ holds. Factor f_1 is $\gamma + 1$, since the cost of reconnecting the edges from S_1 is γ and we have to add the locally minimal edge itself, causing the additional term. Edges from S_2 are inserted into the main tour with factor γ and for presence in H_i, we gain $\beta^2 + \beta$. Factor f_3 follows from Lemma 4. Using that $c(C') \le \sum_{i=1}^{k} c(L_i') \le k \max_{i=1}^{k} c(L_i') \le k\beta^{\log_2(\lfloor 3k/2 \rfloor + 1)} c(S_4)$, we obtain f_4. Then we can estimate the cost of H:

$$\begin{aligned} c(H) &\le (1+\gamma) \cdot c(S_1) + (\beta^2 + \beta + \gamma) \cdot c(S_2) + (\beta^2 + \beta) \cdot c(S_3) + k\gamma \cdot c(S_4) \\ &\le (\beta^2 + \beta + \gamma) \cdot c(S_2) + \max\{1+\gamma, \beta^2 + \beta, k\gamma\} \cdot (c(S_1) + c(S_3) + c(S_4)) \end{aligned}$$

From $\beta > 1$ and $k > 2$ ($k \in \mathbb{N}$), we get $\gamma > 1$, $\max\{1 + \gamma, \beta^2 + \beta, k\gamma\} = k\gamma$, $\gamma > \beta^2 > \beta$, and $\beta^2 + \beta + \gamma \le 3\gamma \le k\gamma$. Thus, we have

$$c(H) \le k\gamma \cdot c(T) \le k\beta^{\log_2(\lfloor 3k/2 \rfloor + 1)} \cdot c(OPT).$$

We can easily build a minimum spanning tree and then search it in breadth-first-search manner in time $\mathcal{O}(|V|^2)$. The paths from step 2 can be constructed sequentially by a search on the tree starting from a special vertex and ending in the consecutive special vertex. The time complexity of such a simple tree traversal is $\mathcal{O}(k \cdot |V|)$. The renaming in step 3 can be done in a linear time, and the ordering can be found by a simple sorting algorithm in time $\mathcal{O}(|V|^2)$. All the following steps until step 10 can be performed by a couple of traversals of the unique paths in time $\mathcal{O}(k \cdot |V|)$. The subtrees T_i ($1 \le i \le q$), its roots r_i and the cheapest edges e_i^* from step 12 can be identified by a linear-time breadth-first-search traversal of the tree. Finally, according to Theorem 1, the procedure HCT^3 (refined) can be performed in quadratic time with respect to the size of each tree. Hence, the overall time complexity of the step 12 for all the trees is $\mathcal{O}(|V|^2)$. The merge in the last step of the algorithm can be also performed in $\mathcal{O}(|V|^2)$ which proves the overall time complexity of our algorithm. □

An obvious question is, whether we can find a better algorithm for $\beta > 1$ by reducing the number of edges skipped in T by a single edge of some L_i, which was bounded by $\lfloor 3k/2 \rfloor$ in Lemma 1. We answer this question to the negative. The following example shows that we cannot always avoid long jumps.

For an even number k, consider the instance that is formed by a path of $2k - 1$ vertices such that the distance between consecutive vertices is 1 and the

distance between the remaining vertices is maximal, considering the β-triangle inequality.

The special vertices are the first $k/2$ and the last $k/2$ of the path. The order is as depicted in Figure 6.

Each of the direct connections between consecutive special vertices has a length of at least $3k/2 - 1$. Therefore, each path connecting two special vertices skipping fewer than $3k/2 - 2$ vertices has to visit one of the $k - 1$ middle vertices. There are, however, k paths that have to bypass the middle vertices. Therefore, by the pigeonhole principle, at least one of them has to jump directly.

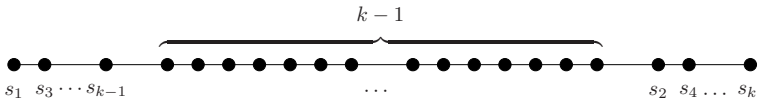

Figure 6: Input instance with long jumps.

5. Conclusion

In this paper, we have studied a variant of TSP where a given subset of vertices appear in prescribed order in the computed Hamiltonian cycle. When the edge costs satisfy the triangle inequality, we have improved the approximation ratio to $2.5 - 2/k$, where k is the number of vertices ordered in the cycle. If, instead of the classical triangle inequality, we consider a relaxed triangle inequality with $\beta > 1$, we have improved the approximation ratio to $k\beta^{\log_2(\lfloor 3k/2 \rfloor + 1)}$. In the latter case, we have given an example showing that jumps of length approximately $3k/2$ cannot be avoided. Note that, even though the edges are long in this example, the achieved approximation ratio is very good. This is the case because also any optimal solution has to contain expensive edges. As an open problem, we would like to pose the question whether we can obtain improved results by giving better bounds on the cost of optimal solutions.

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] Gutin, G., Punnen, A.P. (eds.): The Traveling Salesman Problem and Its Variations. Combinatorial Optimization, Springer, New York (2007)

[6] Micali, S., Vazirani, V. V.: An $\mathcal{O}(\sqrt{(}|V|)|E|)$ Algorithm for Finding Maximum Matching in General Graphs. In: FOCS 1980, pp. 17–27, IEEE Comp. Society (1980)

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

[8] West, D.B.: Introduction to Graph Theory. Prentice Hall Inc., Upper Saddle River, NJ (2000)