Parameterized Approximation Algorithms for TSP

Jianqi Zhou
School of Computer Science and Technology, Shandong University, Qingdao, China

Peihua Li
School of Computer Science and Technology, Shandong University, Qingdao, China

Jiong Guo
School of Computer Science and Technology, Shandong University, Qingdao, China

Abstract

We study the Traveling Salesman problem (TSP), where given a complete undirected graph $G = (V, E)$ with n vertices and an edge cost function $c : E \mapsto R_{\geqslant 0}$, the goal is to find a minimum-cost cycle visiting every vertex exactly once. It is well-known that unless P = NP, TSP cannot be approximated in polynomial time within a factor of $\rho(n)$ for any computable function ρ, while the metric case of TSP, that the edge cost function satisfies the $\triangle$-inequality, admits a polynomial-time 1.5-approximation. We investigate TSP on general graphs from the perspective of parameterized approximability. A parameterized ρ-approximation algorithm returns a ρ-approximation solution in $f(k) \cdot |I|^{O(1)}$ time, where f is a computable function and k is a parameter of the input I. We introduce two parameters, which measure the distance of a given TSP-instance from the metric case, and achieve the following two results:

- A 3-approximation algorithm for TSP in $O((3k_1)!\, 8^{k_1} \cdot n^2 + n^3)$ time, where k_1 is the number of triangles in which the edge costs violate the $\triangle$-inequality.
- A 3-approximation algorithm for TSP in $O(n^{O(k_2)})$ time and a $(6k_2 + 9)$-approximation algorithm for TSP in $O(k_2^{O(k_2)} \cdot n^3)$ time, where k_2 is the minimum number of vertices, whose removal results in a metric graph.

To our best knowledge, the above algorithms are the first non-trivial parameterized approximation algorithms for TSP on general graphs.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Approximation algorithms analysis

Keywords and phrases FPT-approximation algorithms, the Traveling Salesman problem, the triangle inequality, fixed-parameter tractability, metric graphs

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2022.50

Funding The work has been supported by the National Natural Science Foundation of China (No. 61772314, 61761136017, and 62072275).

Acknowledgements The authors thank the reviewers for their valuable comments and constructive suggestions.

1 Introduction

The Traveling Salesman problem (TSP) is one of the most prominent combinatorial optimization problems, where given a complete undirected graph $G = (V, E)$ with n vertices and an edge cost function $c : E \mapsto R_{\geqslant 0}$, the goal is to find a minimum-cost cycle visiting every vertex exactly once. TSP is NP-hard [16, 23]. Bellman [4] and Held and Karp [21] independently proposed an exact algorithm for TSP in $O(2^n)$ time by the dynamic programming technique.

Approximation algorithms. A ρ-approximation algorithm for a minimization problem returns a feasible solution A with $c(A) \leq \rho \cdot c(\text{opt})$ in polynomial time, where opt is an optimal solution. Unfortunately, TSP cannot be approximated in polynomial time within a factor of $\rho(n)$ for any computable function ρ, unless P = NP [34]. There is a special case of TSP called the metric TSP, $\triangle$TSP for short, where $c(u,v) \leq c(u,w) + c(w,v)$ for every $u, v, w \in V$. Hereby, $c(u,v)$ is the cost of the edge between the vertices u and v. $\triangle$TSP is NP-hard [23], but there is a 3/2-approximation algorithm for $\triangle$TSP in $O(n^3)$ time, proposed independently by Christofides [11] and Serdyukov [35]. This has been the best approximation ratio for more than 40 years until recently Karlin, Klein and Gharan [22] gave a randomized $(3/2 - \epsilon)$-approximation algorithm for $\triangle$TSP for some constant $\epsilon > 10^{-36}$. There is a generalization of $\triangle$TSP by relaxing the $\triangle$-inequality to the parameterized $\triangle$-inequality, i.e., for some $\tau \geq 1$, $c(u,v) \leq \tau[c(u,w) + c(w,v)]$ for every $u, v, w \in V$. The parameterized $\triangle$-inequality is denoted by $\triangle_\tau$-inequality and TSP satisfying the $\triangle_\tau$-inequality is denoted by $\triangle_\tau$TSP. Andreae and Bandelt [2] gave a $(3\tau^2 + \tau)/2$-approximation algorithm for $\triangle_\tau$TSP, and then Bender and Chekuri [5] improved the approximation ratio to 4τ. There are some other improved approximation algorithms for $\triangle_\tau$TSP with $\tau \leq 13/3$, e.g., [1, 6, 30]. These approximation results represent no conflict with the polynomial-time inapproximability of TSP, since τ is not a constant and even not a function of n. Mohan [29] designed a 7/2-approximation algorithm for Biased-TSP, where all vertices can be partitioned into two parts such that every triangle with three vertices in the same part satisfies the $\triangle$-inequality, while the triangles with vertices in different parts may violate the $\triangle$-inequality. However, not all TSP-instances admit such a partition.

Parameterized algorithms. An exact algorithm with running time of the form $f(k) \cdot |I|^{O(1)}$ is called a parameterized algorithm and a problem admitting a parameterized algorithm is fixed-parameter tractable (FPT). The idea is to limit the exponential running time to the parameter k instead of the input size $|I|$. Deineko and Hoffmann [12] studied TSP in the 2-dimensional Euclidean plane called 2DTSP that is NP-hard [32], and gave a parameterized algorithm in $O(2^k k^2 \cdot n)$ time for 2DTSP with k inner points, where a point is inner if it lies strictly inside the convex hull of the input. This was then improved to $O(k^{11\sqrt{k}} k^{1.5} \cdot n^3)$ by Knauer and Spillner [26].

Parameterized approximation algorithms. A parameterized ρ-approximation algorithm returns a ρ-approximation solution in $f(k) \cdot |I|^{O(1)}$ time, where f is a computable function and k is a parameter of the input I. Many prominent NP-hard problems have been studied from the perspective of parameterized approximability, for instance, Independent Set [8, 13, 18, 19, 27, 28] and Dominating Set [9, 10, 33]. Concerning TSP, Arora [3] proposed a $(1+\epsilon)$-approximation algorithm for TSP in the k-dimensional Euclidean space running in $O(n(\log n)^{O(\sqrt{k}/\epsilon)^{k-1}})$ time for any $\epsilon \geq 0$, which can be upper bounded by $O(k^{O(\sqrt{k}/\epsilon)^{k-1}} \cdot n^2)$ as shown in [15, 24]. Some parameterized approximation algorithms for the metric TSP with other dimensional parameters are introduced in [15, 17]. Bockenhauer and Hromkovic [7] studied TSP with deadlines, DLTSP for short, where k vertices must be visited before a given time. DLTSP cannot be approximated in polynomial time within a factor of $\rho(n)$ for any polynomial function ρ. Bockenhauer and Hromkovic [7] gave a parameterized 5/2-approximation algorithm for the metric DLTSP running in $O(k!\, k + n^3)$ time, where k is the number of deadline vertices. We refer the readers to the survey [15, 28] of the parameterized approximation algorithms.

Our contributions. We aim at designing parameterized approximation algorithms for TSP on general graphs. Motivated by the polynomial-time inapproximability of TSP and the 3/2-approximation algorithm for $\triangle$TSP, we propose a concept for designing parameterized approximation algorithms, the so-called “distance from approximability”, that is, introducing a parameter measuring the distance between the inapproximable and approximable cases and designing approximation algorithms with the exponential running time restricted to the parameter. A similar concept called “distance from triviality” has been used in the parameterized algorithmics [20]. Here, we introduce two parameters to measure the distance between a general TSP-instance and a metric TSP-instance. First, we consider the number k_1 of triangles in a general TSP-instance, which do not satisfy the $\triangle$-inequality, that is, the triangles in which the edge costs violate the $\triangle$-inequality. Given a TSP-instance, the value of k_1 can be easily computed in $O(n^3)$ time. Using k_1 as parameter, we achieve an approximation of TSP with an approximation factor of 3 and a running time of $O((3k_1)!\,8^{k_1} \cdot n^2 + n^3)$. Our second parameter k_2 is set equal to the minimum number of vertices, whose removal turns a general instance into a metric instance. Clearly, we can apply a search tree [14, 31] to determine the value of k_2 in $O(3^{k_2} \cdot n^3)$ time. Moreover, we always have $k_2 \leq k_1$. With k_2 as parameter, we first present an algorithm running in $O(n^{O(k_2)})$ time, returning a TSP-solution with an approximation factor of 3 and then a parameterized approximation algorithm with a factor of $6k_2$+9 and a running time of $O(k_2^{O(k_2)} \cdot n^3)$. To our best knowledge, the above algorithms are the first non-trivial parameterized approximation algorithms for general TSP. We are also confident that the “distance from approximability” concept might be useful for other problems.

2 Preliminaries

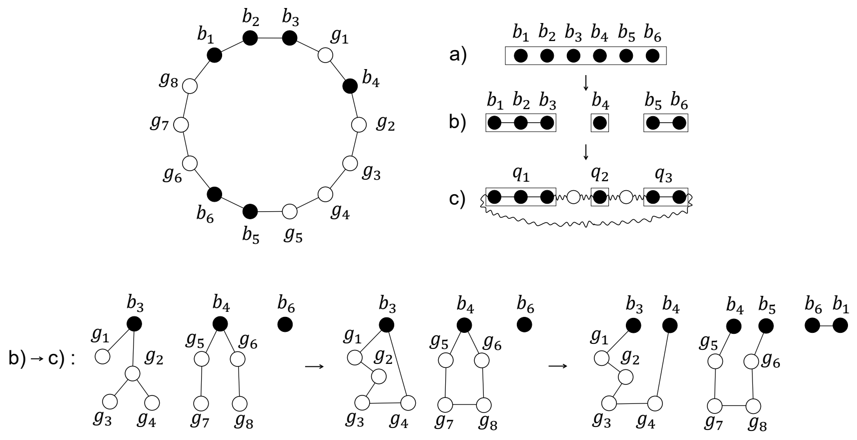
We introduce some basic definitions and notations. We consider a simple undirected complete graph $G = (V, E)$, with an edge cost function $c : E \mapsto R_{\geqslant 0}$. The graph G is metric if $c(u, v) \leq c(u, w) + c(w, v)$ for every $u, v, w \in V$. A triangle $\triangle(u, v, w)$ is called “violating” if it does not satisfy the $\triangle$-inequality, that is, at least one of $c(u, v) \leq c(u, w) + c(w, v)$, $c(u, w) \leq c(u, v) + c(v, w)$ and $c(v, w) \leq c(v, u) + c(u, w)$ does not hold. A set of vertices whose removal transforms a non-metric graph into a metric graph, is called a violating vertex set. For an edge set $E' \subseteq E$, the cost of E', denoted by $c(E')$, is the total cost of its edges, and the vertex set of E' is denoted by $V(E')$. For a vertex set $V' \subseteq V$, $E(V')$ is the set of edges that connect two vertices in V', and $G[V'] = (V', E(V'))$ is the subgraph of G induced by V'. For a positive integer i, set $[i] = \{1, \cdots, i\}$.

An acyclic graph consisting of t connected components is called a t-forest. A spanning subgraph of G, which is a t-forest, is called a spanning t-forest of G. A spanning t-forest having the minimum cost is called a t-minimum spanning forest (t-MSF) of G.

3 The number of violating triangles as parameter

In this section, we consider a TSP-instance with k_1 triangles violating the $\triangle$-inequality. The parameter k_1 can be computed in $O(n^3)$ time by checking all triangles in the input. We call a vertex “bad” if it is in one of the k_1 triangles, and we denote by V^b the set of all bad vertices, $|V^b| \leq 3k_1$. The remaining vertices are called “good” and we denote by V^g the set of good vertices. For $b_u, b_v, b_w \in V^b$, the triangle formed by these three vertices $\triangle(b_u, b_v, b_w)$ might violate the $\triangle$-inequality, that is, $c(b_u, b_w) + c(b_w, b_v)$ might be arbitrarily less than $c(b_u, b_v)$.

Our algorithm consists mainly of two steps. The first step “guesses” the occurrences of bad vertices in an optimal TSP-solution, that is, the occurrence order of bad vertices and the “gaps” between bad vertices, where the good vertices should be inserted to form the



b_1 b_2 b_3 b_4 b_5 b_6
g_1 g_2 g_3 g_4 g_5 g_6 g_7 g_8
a)
b)
c)
q_1 q_2 q_3
b) → c) :

Figure 1 An illustration of Algorithm 1. b_i's are the bad vertices, g_i's are the good vertices and q_i's are the subsets of the bad vertices. Top left: the cycle represents an optimal TSP-solution. Top right: a) a permutation of bad vertices; b) a partition of bad vertices respecting the order of the permutation in a); c) inserting good vertices into the gaps between bad vertices. Bottom: the first graph is the MSF rooted at the end-vertices of the subsets of the partition; the second graph is a collection of cycles (and possibly isolated vertices) obtained by doubling the edges of the MSF in the first graph; the third graph is the paths with only good internal vertices connecting consecutive subsets (and possibly edges directly connecting consecutive subsets).

optimal TSP-solution. This can be done by enumerating all possible permutations of bad vertices and for each permutation, all partitions of bad vertices, which respect the order of the permutation. This means that the bad vertices in one subset of the partition occur together in the optimal TSP-solution and obey the order of the permutation. The subsets also occur in the order of the permutation. The optimal TSP-solution contains at least one good vertex between two consecutive subsets. The second step computes paths of good vertices to fill in the gaps between the subsets in each partition of permutations. Hereby, we compute the minimum-cost spanning forest rooted at the end-vertices of the subsets of the partition and transform it to a collection of paths with only good internal vertices (and possibly edges directly connecting consecutive subsets). See Figure 1 for an illustration.

We introduce some notations to describe the bad vertices. A "bad chain" denoted by $q = (b_1, b_2, \ldots, b_l)$ with $l \geq 1$ is one bad vertex or a path consisting of distinct bad vertices, where $b_i \in V^b$ for $i \in [l]$ and there is an edge connecting b_i and b_{i+1} for each $i \in [l-1]$. We use $b_s(q)$, $b_e(q)$ to denote the starting and ending vertices, respectively, and use $c(q)$ to denote the total cost of the edges in q. In particular, if a bad chain q consists of only one bad vertex b_1, then $b_s(q) = b_1$, $b_e(q) = b_1$, and $c(q) = 0$.

3.1 The algorithm and time complexity

The algorithm is as follows (see Figure 1 for an illustration).

Algorithm 1 Ratio-3 approximation algorithm for parameterization with k_1.

1. Compute the k_1 triangles violating the $\triangle$-inequality, the set of bad vertices V^b and the set of good vertices V^g.
2. Enumerate all possible permutations of bad vertices. For each permutation, enumerate all possible partitions of bad vertices into t subsets for each $t \in [|V^b|]$, respecting the corresponding permutation order. For each t-partition, do the following:

(a) Connect the bad vertices in each subset of the partition in the order of the permutation, resulting in t bad chains $Q = (q_1, \dots, q_t)$, ordered according to their orders in the permutation.
(b) Compute a t-minimum spanning forest (t-MSF) $F = (T_1, \dots, T_t)$ of $G[V^g \cup \{b_e(q_i) | i \in [t]\}]$ rooted at $\{b_e(q_i) | i \in [t]\}$.
(c) Double the edges of T_i's. Compute an Euler tour of each T_i and shortcut repeated vertices in the Euler tour, resulting in C_i.
(d) For each $i \in [t]$, do the following: if $C_i = (b_e(q_i))$, then define $p_i = (b_e(q_i), b_s(q_{i+1}))$; if $C_i = (b_e(q_i), g_{i_1}, \dots, g_{i_{l_i}}, b_e(q_i))$ with $l_i \geq 1$, then define $p_i = (b_e(q_i), g_{i_1}, \dots, g_{i_{l_i}}, b_s(q_{i+1}))$. Hereby, $q_{t+1} = q_1$.
(e) Connect $b_e(q_i)$ and $b_s(q_{i+1})$ in this t-partition by p_i for each $i \in [t]$, and obtain a TSP-solution $A = (q_1, p_1, q_2, p_2, \dots, q_t, p_t)$.

3. Return the solution $A^{\min}$ with the minimum cost among all enumeration cases.

▶ **Lemma 1.** *A minimum spanning forest of $G[V \cup V']$ rooted at V' can be computed in $O(|V \cup V'|^2)$ time.*

Proof. First, we introduce a root vertex and connect it to all vertices in V' by zero-cost edges to get a new graph $\tilde{G}$, transforming computing a minimum spanning forest rooted at V' into computing a minimum spanning tree (MST).

An MST T of $\tilde{G}$ can be found in $O(|V(\tilde{G})|^2)$ time. If T does not contain all edges connecting the root vertex to the vertices in V', then we can add those missing edges, remove other edges incident to the vertices in V' to ensure acyclic, and obtain a new MST T' without increasing costs. By removing the root vertex and all edges incident to it in T', we get a minimum spanning forest of $G[V \cup V']$ rooted at V' in $O(|V \cup V'|^2)$ time. ◀

▶ **Lemma 2.** *Algorithm 1 runs in $O((3k_1)!\, 8^{k_1} \cdot n^2 + n^3)$ time.*

Proof. Step 1 computes k_1 triangles violating the $\triangle$-inequality in $O(n^3)$ time. The number of t-partitions of permutations enumerated in Step 2 is $|V^b|!\, 2^{|V^b|} = O((3k_1)!\, 8^{k_1})$. For each possible t-patition, Step 2(a) to Step 2(d) take $O(n^2)$ time by Lemma 1. Hence, Algorithm 1 runs in $O((3k_1)!\, 8^{k_1} \cdot n^2 + n^3)$ time. ◀

3.2 Analysis of approximation factor

An optimal TSP-solution opt can be decomposed into an ordered collection of $2t^{\text{opt}}$ many paths $q_1^{\text{opt}}, p_1^{\text{opt}}, q_2^{\text{opt}}, \dots, p_{t^{\text{opt}}}^{\text{opt}}$, with q_i^{opt} being a bad chain and $p_i^{\text{opt}} = (b_e(q_i^{\text{opt}}), g_{i_1}^{\text{opt}}, \dots, g_{i_{l_i^{\text{opt}}}}^{\text{opt}}, b_s(q_{i+1}^{\text{opt}}))$ being the path connecting $b_e(q_i^{\text{opt}})$ and $b_s(q_{i+1}^{\text{opt}})$ with only good internal vertices. Here $l_i^{\text{opt}} \geq 1$, $q_{t^{\text{opt}}+1}^{\text{opt}} = q_1^{\text{opt}}$. Clearly, $t^{\text{opt}} \leq 3k_1$. See Figure 1 for an example: $t^{\text{opt}} = 3$, $q_1^{\text{opt}} = (b_1, b_2, b_3)$, $q_2^{\text{opt}} = (b_4)$, $q_3^{\text{opt}} = (b_5, b_6)$. Moreover, $p_1^{\text{opt}} = (b_3, g_1, b_4)$ where $l_1^{\text{opt}} = 1$, connects $b_e(q_1) = b_3$ and $b_s(q_2) = b_4$. Further, $p_2^{\text{opt}} = (b_4, g_2, g_3, g_4, g_5, b_5)$ and $p_3^{\text{opt}} = (b_6, g_6, g_7, g_8, b_1)$. Then, $c(\text{opt}) = \sum_{i=1}^{t^{\text{opt}}} c(q_i^{\text{opt}}) + \sum_{i=1}^{t^{\text{opt}}} c(p_i^{\text{opt}})$.

We enumerate all partitions of every permutation of bad vertices in Step 2 of Algorithm 1. Thus, there must exist an enumeration case α in Step 2, where we have t^{opt} bad chains which contain the same bad vertices in the same order as in opt. Therefore, the solution A^{α} returned in Step 2(e) of this case differs from opt only in the paths connecting the bad chains. In the following, we give an analysis of these paths for both opt and A^{α}.

▶ **Lemma 3.** *For each $i \in [t^{opt}]$, $c(p_i^{opt}) \geq c(b_e(q_i^{opt}), b_s(q_{i+1}^{opt}))$, where $q_{t^{opt}+1}^{opt} = q_1^{opt}$.*

Proof. Note that a triangle containing at least one good vertex satisfies the $\triangle$-inequality. Thus, $\triangle(b_e(q_i^{\text{opt}}), g_{i_1}^{\text{opt}}, b_s(q_{i+1}^{\text{opt}}))$ and $\triangle(g_{i_1}^{\text{opt}}, g_{i_{l_i^{\text{opt}}}}^{\text{opt}}, b_s(q_{i+1}^{\text{opt}}))$ satisfy the $\triangle$-inequality. Here $l_i^{\text{opt}} \geq 1$, $q_{t^{\text{opt}}+1}^{\text{opt}} = q_1^{\text{opt}}$. Thus, for each $i \in [t^{\text{opt}}]$, we have

$$\begin{aligned}
c(p_i^{\text{opt}}) &= c(b_e(q_i^{\text{opt}}), g_{i_1}^{\text{opt}}) + c(g_{i_1}^{\text{opt}}, \dots, g_{i_{l_i^{\text{opt}}}}^{\text{opt}}) + c(g_{i_{l_i^{\text{opt}}}}^{\text{opt}}, b_s(q_{i+1}^{\text{opt}})) \\
&\geq c(b_e(q_i^{\text{opt}}), g_{i_1}^{\text{opt}}) + c(g_{i_1}^{\text{opt}}, g_{i_{l_i^{\text{opt}}}}^{\text{opt}}) + c(g_{i_{l_i^{\text{opt}}}}^{\text{opt}}, b_s(q_{i+1}^{\text{opt}})) \\
&\geq c(b_e(q_i^{\text{opt}}), g_{i_1}^{\text{opt}}) + c(g_{i_1}^{\text{opt}}, b_s(q_{i+1}^{\text{opt}})) \\
&\geq c(b_e(q_i^{\text{opt}}), b_s(q_{i+1}^{\text{opt}})).
\end{aligned}$$ ◀

In Step 2(b) of the case α, we obtain a t^{opt}-MSF $F^\alpha = (T_1^\alpha, \dots, T_{t^{\text{opt}}}^\alpha)$ of $G[V^g \cup \{b_e(q_i^{\text{opt}}) | i \in [t^{\text{opt}}]\}]$ rooted at $\{b_e(q_i^{\text{opt}}) | i \in [t^{\text{opt}}]\}$. Then, let C_i^α for $i \in [t^{\text{opt}}]$ be the resulting cycle (or the isolated vertex) in Step 2(c) of the case α. Let p_i^α for $i \in [t^{\text{opt}}]$ be the path returned in Step 2(d) of the case α. Then $c(A^\alpha) = \sum_{i=1}^{t^{\text{opt}}} c(q_i^{\text{opt}}) + \sum_{i=1}^{t^{\text{opt}}} c(p_i^\alpha)$. The following lemma establishes the relation of the cost of p_i^α with the cost of p_i^{opt}.

▶ **Lemma 4.** *For each $i \in [t^{opt}]$, $c(p_i^\alpha) \leq c(p_i^{opt}) + 2c(T_i^\alpha)$.*

Proof. For each $i \in [t^{\text{opt}}]$, if $C_i^\alpha = (b_e(q_i^{\text{opt}}))$, then as defined in Step 2(d) of the case α, $p_i^\alpha = (b_e(q_i^{\text{opt}}), b_s(q_{i+1}^{\text{opt}}))$, thus $c(p_i^\alpha) \leq c(p_i^{\text{opt}})$ by Lemma 3.
Otherwise, $C_i^\alpha = (b_e(q_i^{\text{opt}}), g_{i_1}^\alpha, \dots, g_{i_{l_i^\alpha}}^\alpha, b_e(q_i^{\text{opt}}))$ with $l_i^\alpha \geq 1$, Step 2(d) of the case α defines $p_i^\alpha = (b_e(q_i^{\text{opt}}), g_{i_1}^\alpha, \dots, g_{i_{l_i^\alpha}}^\alpha, b_s(q_{i+1}^{\text{opt}}))$. Note that a triangle containing at least one good vertex satisfies the $\triangle$-inequality, and as shown in Lemma 3, we have

$$\begin{aligned}
c(p_i^\alpha) &= c(b_e(q_i^{\text{opt}}), g_{i_1}^\alpha, \dots, g_{i_{l_i^\alpha}}^\alpha) + c(g_{i_{l_i^\alpha}}^\alpha, b_s(q_{i+1}^{\text{opt}})) \\
&\leq c(b_e(q_i^{\text{opt}}), g_{i_1}^\alpha, \dots, g_{i_{l_i^\alpha}}) + c(g_{i_{l_i^\alpha}}^\alpha, b_e(q_i^{\text{opt}})) + c(b_e(q_i^{\text{opt}}), b_s(q_{i+1}^{\text{opt}})) \\
&= c(C_i^\alpha) + c(b_e(q_i^{\text{opt}}), b_s(q_{i+1}^{\text{opt}})) \\
&\leq 2c(T_i^\alpha) + c(p_i^{\text{opt}}).
\end{aligned}$$ ◀

Next, we compare the cost of opt with the cost of F^α, the following lemma is easy to see.

▶ **Lemma 5.** *$c(opt) \geq c(F^\alpha)$.*

Proof. We remove all bad vertices not in $\{b_e(q_i^{\text{opt}}) | i \in [t^{\text{opt}}]\}$ and all edges incident to them from opt. Then for each $i \in [t^{\text{opt}}]$, if q_i^{opt} consists of only one bad vertex, then we remove the edge connecting it and its preceding good vertex. The above removal process results in a spanning t^{opt}-forest of $G[V^g \cup \{b_e(q_i^{\text{opt}}) | i \in [t^{\text{opt}}]\}]$ rooted at $\{b_e(q_i^{\text{opt}}) | i \in [t^{\text{opt}}]\}$. Thus, $c(\text{opt}) \geq c(F^\alpha)$. ◀

Finally, we arrive at proving the main result of this section.

▶ **Theorem 6.** *Algorithm 1 is a parameterized 3-approximation algorithm for TSP running in $O((3k_1)!\, 8^{k_1} \cdot n^3)$ time, where k_1 is the number of triangles in which edge costs violate the $\triangle$-inequality.*

Proof. Algorithm 1 returns $A^{\min}$ in $O((3k_1)!\, 8^{k_1} \cdot n^2 + n^3)$ time by Lemma 2 and clearly provides a cycle over all vertices. The approximation factor follows from Lemmas 4 and 5: $c(A^{\min}) \leq c(A^\alpha) = \sum_{i=1}^{t^{\text{opt}}} c(q_i^{\text{opt}}) + \sum_{i=1}^{t^{\text{opt}}} c(p_i^\alpha) \leq \sum_{i=1}^{t^{\text{opt}}} c(q_i^{\text{opt}}) + \sum_{i=1}^{t^{\text{opt}}} c(p_i^{\text{opt}}) + 2\sum_{i=1}^{t^{\text{opt}}} c(T_i^\alpha) = c(\text{opt}) + 2c(F^\alpha) \leq c(\text{opt}) + 2c(\text{opt}) = 3c(\text{opt})$. ◀

4 The minimum size of violating vertex sets as parameter

In this section, we consider the minimum size k_2 of violating vertex sets, that is, the minimum number of vertices whose removal transforms a given graph into a metric graph. The violating vertex set with at most k_2 vertices can be found in $O(3^{k_2} \cdot n^3)$ time by the search tree technique [14, 31]. The basic idea is that at least one vertex of every triangle violating the $\triangle$-inequality has to be added to the violating vertex set, which also implies that $k_2 \leq k_1$. We use the bad vertices to refer to the vertices in the violating vertex set and thus, there are k_2 bad vertices. The remaining vertices are called good. A bad chain also consists solely of bad vertices.

In contrast to Section 3, a triangle violating the $\triangle$-inequality might contain only one or two bad vertices, leading to the consequence that the idea of Algorithm 1 does not directly apply to the parameterization of k_2. In Section 3, triangles containing at least one good vertex satisfy the $\triangle$-inequality, which provides the foundation of Lemmas 3 and 4. Notice that, as shown in Lemma 3, the cost of the edge directly connecting two consecutive bad chains in opt does not exceed the cost of the path in opt connecting the same two bad chains. Here, this property does not hold. However, by a closer observation, we can conclude that a path connecting bad chains only involves the starting and ending bad vertices of bad chains. If we can fix the two good vertices, which are the direct neighbors of the bad chains in the optimal TSP-solution, then we can extend bad chains to paths, which are between two good vertices and whose internal vertices are all bad vertices. Then, we use the same strategy as in Steps 2(b)-2(d) in Algorithm 1 to find paths of good vertices to connect these paths, where we deal only with edges between good vertices, to which we can apply the analysis in Section 3. We show in the following that a brute-force way of fixing the direct neighbors of bad chains, which is Algorithm 2, leads to a 3-approximation with the running time of $O(n^{O(k_2)})$, while a more involved selection of the direct neighbors, i.e., Algorithm 3, gives a parameterized approximation with the running time of $O(k_2^{O(k_2)} \cdot n^3)$ but a worse approximation factor $6k_2 + 9$.

To this end, we need the following notations. An "alternating chain" denoted by $h = (g_1, q_1, g_2, \ldots, g_l, q_l, g_{l+1})$ with $l \geq 1$, is a path consisting alternatively of distinct good vertices and bad chains. Here, $g_1, \ldots, g_{l+1} \in V^g$, and $q_1, \ldots, q_l$ are vertex-disjoint bad chains. Moreover, g_{i+1} for $i \in [l-1]$ is connected by edges to $b_e(q_i)$ and $b_s(q_{i+1})$, g_1 is adjacent to $b_s(q_1)$, and g_{l+1} is adjacent to $b_e(q_l)$. We use $g_s(h)$, $g_e(h)$ to denote the starting and ending good vertices, respectively, and use $c(h)$ to denote the total cost of the edges in h, i.e., $c(h) = \sum_{i=1}^{l} c(q_i) + \sum_{i=1}^{l} c(g_i, b_s(q_i)) + \sum_{i=1}^{l} c(b_e(q_i), g_{i+1})$.

4.1 A 3-approximation algorithm

In this section, we use a brute-force strategy to fix the direct good neighbors of the bad chains in the optimal TSP-solution.

Algorithm 2 Ratio-3 approximation algorithm for parameterization with k_2.

1. Compute the violating vertex set of k_2 vertices V^b by the search tree technique and the set of good vertices V^g.
2. Enumerate all possible permutations of bad vertices. For each permutation, enumerate all possible partitions of bad vertices into t_1 subsets for each $t_1 \in [k_2]$, respecting the corresponding permutation order. For each t_1-partition, do the following:
   - **(a)** Connect the bad vertices in each subset of the partition in the order of the permutation, resulting in t_1 bad chains $Q = (q_1, \ldots, q_{t_1})$, ordered according to their orders in the permutation.

(b) For all possible t_1-tuples of pairs of good vertices $((x_1, y_1), (x_2, y_2), \dots, (x_{t_1}, y_{t_1}))$ satisfying $x_i, y_i \in V^g \backslash \bigcup_{j=1}^{i-1}\{x_j, y_j\}$ for each $i \in [t_1]$: (Note that it might be $x_i = y_i$ for some i.)

(i) Transform q_i for each $i \in [t_1]$ to an alternating chain $h_i = (y_{i-1}, q_i, x_i)$. Here, $y_0 = y_{t_1}$. If $x_i = y_i$, then merge two alternating chains h_i and h_{i+1} into one, resulting in $H = (h_1, \dots, h_{t_2})$ with $t_2 \le t_1$ and no two consecutive alternating chains being mergeable.

(ii) Apply the following FindPaths procedure.
Procedure FindPaths:

(1) Compute a t_2-minimum spanning forest (t_2-MSF) $F = (T_1, \dots, T_{t_2})$ of $G[V^g \backslash \bigcup_{j=1}^{t_2}(V(h_j)\backslash\{g_e(h_j)\})]$ rooted at $\{g_e(h_j)|j \in [t_2]\}$.

(2) Double the edges of T_j's. Compute an Euler tour of each T_j and shortcut repeated vertices in the Euler tour, resulting in C_j.

(3) For each $j \in [t_2]$, do the following: if $C_j = (g_e(h_j))$, then define $p_j = (g_e(h_j), g_s(h_{j+1}))$; if $C_j = (g_e(h_j), g_{j_1}, \dots, g_{j_{l_j}}, g_e(h_j))$ with $l_j \ge 1$, then define $p_j = (g_e(h_j), g_{j_1}, \dots, g_{j_{l_j}}, g_s(h_{j+1}))$. Hereby, $h_{t_2+1} = h_1$.

(4) Connect $g_e(h_j)$ and $g_s(h_{j+1})$ in this t_2-partition by p_j for each $j \in [t_2]$, and obtain a TSP-solution $A = (h_1, p_1, h_2, p_2, \dots, h_{t_2}, p_{t_2})$.

3. Return the solution $A^{\min}$ with the minimum cost among all enumeration cases.

The running time of this algorithm is dominated by the enumeration of the tuples of pairs of good vertices in Step 2(b).

▶ **Lemma 7.** *Algorithm 2 runs in $O(n^{O(k_2)})$ time.*

Proof. Step 1 computes k_2 bad vertices by search tree technique in $O(3^{k_2} \cdot n^3)$ time. The number of t_1-partitions of permutations enumerated in Step 2 is the same as in Lemma 2, $k_2!\, 2^{k_2}$. For each possible t_1-partition, the number of the t_1-tuples enumerated in Step 2(b) is $O(n^{2t_1}) = O(n^{2k_2})$. By the same argument as in Lemma 1, Algorithm 2 needs $O(n^2)$ time in Step 2(b)(ii). Hence, Algorithm 2 runs in $O(3^{k_2} \cdot n^3 + k_2!\, 2^{k_2} \cdot n^{2k_2} \cdot n^2) = O(n^{O(k_2)})$ time. ◀

Given an ordered collection of vertex-disjoint alternating chains $H = (h_1, \dots, h_{t_2})$ containing all bad vertices, the FindPaths procedure deals only with edges between good vertices, to which we can apply the similar analysis as in Lemmas 3-5 and Theorem 6. Let opt^H be a minimum-cost TSP-solution, where the vertices in $\bigcup_{i=1}^{t_2} V(h_i)$ occur in the same alternating chains with the same order as specified in H. The following lemma is easy to see.

▶ **Lemma 8.** *Given an ordered collection of vertex-disjoint alternating chains $H = (h_1, \dots, h_{t_2})$ containing all bad vertices, the FindPaths procedure returns a TSP-solution A with $c(A) \le 3c(opt^H)$.*

We obtain the main result of this subsection as a consequence of Lemmas 7 and 8.

▶ **Theorem 9.** *Algorithm 2 is a 3-approximation algorithm for TSP running in $O(n^{O(k_2)})$ time, where k_2 is the minimum number of vertices whose removal results in a metric graph.*

Proof. Algorithm 2 returns $A^{\min}$ in $O(n^{O(k_2)})$ time by Lemma 7 and the output $A^{\min}$ is clearly a cycle over all vertices. Given an optimal TSP-solution opt, let Q^{opt} denote the ordered collection of all bad chains in opt, and H^{opt} denote the corresponding ordered collection of all alternating chains in opt. By the same analysis as in Section 3, there is a case enumerated in Step 2, which gives the same order of bad chains as Q^{opt} in Step 2(a). Since good vertices occur only once in opt, there must be case α enumerated in Step 2(b), where we have the same alternating chains and they are in the same order as H^{opt} in Step 2(b)(i). By Lemma 8, in this case, the FindPaths procedure returns a TSP-solution A^α with $c(A^\alpha) \le 3c(\text{opt}^{H^{\text{opt}}}) = 3c(\text{opt})$. Thus, $c(A^{\min}) \le c(A^\alpha) \le 3c(\text{opt})$. ◀

4.2 A parameterized $(6k_2 + 9)$-approximation algorithm

In this section, we apply a more involved strategy to fix the good vertices, that are direct neighbors of the bad chains in the optimal TSP-solution. We give an upper bound $f(k_2)$ on the number of possible good neighbors with f being a polynomial function. To achieve this, we slightly modify Algorithm 2. Given t_1 bad chains, we divide all good vertices into t_1 connected components with minimum edge costs by computing a t_1-minimum spanning forest. For each bad chain, we choose some good vertices from each connected component to form a set of good vertices that may be direct neighbors of the bad chain.

4.2.1 The algorithm and time complexity

The Step 1 and Step 2(a) of the parameterized approximation algorithm (Algorithm 3) are the same as Algorithm 2. Thus, we skip them in the description of Algorithm 3.

Algorithm 3 Ratio-$6k_2 + 9$ approximation algorithm for parameterization with k_2.

```
2. (b) Use Khachay and Neznakhina's algorithm [25] to get a $t_1$-minimum spanning forest
       of $G[V^g]$, $\mathcal{F} = (\mathcal{T}_1, \ldots, \mathcal{T}_{t_1})$. Let $V_j = V(\mathcal{T}_j)$ for $j \in [t_1]$.
       For each bad vertex $b$, let $N(b, V_j)$ be the set of $\min\{4t_1, |V_j|\}$ good vertices in $V_j$,
       which are closest to $b$. That is, $\forall g' \in V_j \backslash N(b, V_j)$ and $\forall g \in N(b, V_j)$, $c(g', b) \ge c(g, b)$.
       Set $N(b) = \bigcup_{j=1}^{t_1} N(b, V_j)$.
   (c) For all possible $t_1$-tuples of pairs of good vertices $((x_1, y_1), (x_2, y_2), \ldots, (x_{t_1}, y_{t_1}))$
       satisfying $x_i \in N(b_e(q_i)) \backslash \bigcup_{j=1}^{i-1} \{x_j, y_j\}$ and $y_i \in N(b_s(q_{i+1})) \backslash \bigcup_{j=1}^{i-1} \{x_j, y_j\}$ for each
       $i \in [t_1]$: (Note that it might be $x_i = y_i$ for some $i$ and $q_{t_1+1} = q_1$.)
       (i) Transform $q_i$ for each $i \in [t_1]$ to an alternating chain $h_i = (y_{i-1}, q_i, x_i)$. Here,
           $y_0 = y_{t_1}$. If $x_i = y_i$, then merge two alternating chains $h_i$ and $h_{i+1}$ into one,
           resulting in $H = (h_1, \ldots, h_{t_2})$ with $t_2 \le t_1$ and no two consecutive alternating
           chains being mergeable.
       (ii) Apply the FindPaths procedure.
3. Return the solution $A^{\min}$ with the minimum cost among all enumeration cases.
```

▶ **Lemma 10.** *Algorithm 3 runs in $O(k_2^{O(k_2)} \cdot n^3)$ time.*

Proof. As for Algorithm 2, Step 1 needs $O(3^{k_2} \cdot n^3)$ time. The number of t_1-partitions of permutations enumerated in Step 2 is $k_2!\,2^{k_2}$. For each possible t_1-partition, a t_1-MSF of the graph $G[V^g]$ can be computed in $O(n^2 \log n)$ time [25]. The number of t_1-tuples enumerated in Step 2(c) is bounded by $O((t_1 \cdot 4t_1)^{2t_1}) = O((2k_2)^{4k_2})$. FindPaths needs $O(n^2)$ time. Hence, Algorithm 3 runs in $O(3^{k_2} \cdot n^3 + k_2!\,2^{k_2} \cdot (n^2 \log n + (2k_2)^{4k_2} \cdot n^2)) = O(k_2^{O(k_2)} \cdot n^3)$ time. ◀

4.2.2 Analysis of approximation factor

Again, for an optimal TSP-solution opt with an order of bad chains, $Q^{\text{opt}} = (q_1^{\text{opt}}, \ldots, q_{t_1^{\text{opt}}}^{\text{opt}})$, we have a case α in Step 2 of Algorithm 3 with the same order of bad chains. In Step 2(b) of this case α, we obtain a t_1^{opt}-MSF $\mathcal{F}^\alpha = (\mathcal{T}_1^\alpha, \ldots, \mathcal{T}_{t_1^{\text{opt}}}^\alpha)$ of the graph $G[V^g]$, and $V_j^\alpha = V(\mathcal{T}_j^\alpha)$ for $j \in [t_1^{\text{opt}}]$. The following lemma is easy to see.

▶ **Lemma 11.** *$c(opt) \geq c(\mathcal{F}^\alpha)$.*

Proof. We remove all bad vertices and all edges incident to them from opt, resulting in a spanning t_1^{opt}-forest of the graph $G[V^g]$. Thus, $c(\text{opt}) \geq c(\mathcal{F}^\alpha)$. ◀

The main difficulty with the analysis of Algorithm 3, compared with Algorithm 2, lies in that the order of alternating chains H^{opt} might not be enumerated in Step 2(c). Therefore, we have to adopt a different approach. Hereby, we modify opt to construct another TSP-solution $\tilde{A}$, which satisfies on the one hand $c(\tilde{A}) = O(k_2 \cdot c(\text{opt}))$, and whose order of alternating chains on the other hand occurs in the case α' enumerated by Step 2(c). Thus, we know from Lemma 8 that Step 2(c)(ii) returns a solution $A^{\alpha'}$ in the case α' with $c(A^{\alpha'}) \leq 3c(\tilde{A})$, which completes the proof of approximation ratio.

Construction of opt* from opt. The construction of $\tilde{A}$ consists of three steps. The first step constructs a cycle opt* from opt. For each $j \in [t_1^{\text{opt}}]$, we partition $V_j^\alpha = V(\mathcal{T}_j^\alpha)$ into three disjoint subsets X_j^1, X_j^2 and Y_j. The subset X_j^1 contains the good vertices in V_j^α, each of which is adjacent to one bad vertex and one good vertex in opt. The subset X_j^2 contains the good vertices in V_j^α, each of which is adjacent to two bad vertices in opt. The subset Y_j contains the good vertices in V_j^α, each of which is adjacent to two good vertices in opt. We aim to remove the vertices in Y_j's to get opt*. If $Y_j \neq \emptyset$ and $X_j^1 \neq \emptyset$ for $j \in [t_1^{\text{opt}}]$, then shortcut in opt all vertices in Y_j. If $Y_j \neq \emptyset$ and $X_j^1 = \emptyset$, then pick an arbitrary vertex in Y_j, say y_j, and shortcut in opt all vertices in $Y_j \backslash \{y_j\}$. After completing the above operation for all $j \in [t_1^{\text{opt}}]$, we get a new cycle, denoted by opt*.

Since we shortcut no bad vertex and no good vertex adjacent to bad vertices in opt, we have $c(\text{opt}^*) \leq c(\text{opt})$. The vertices in $X_j^1 \cup X_j^2$ and the possibly existing vertices in Y_j in opt* still keep their properties in opt: each good vertex in X_j^1 is adjacent to one bad vertex and one good vertex in opt*, each good vertex in X_j^2 is adjacent to two bad vertices in opt*, and if exists, y_j is adjacent to two good vertices in opt*.

Next, we partition X_j^1 into two disjoint subsets Z_j^0 and Z_j^1. The subset Z_j^0 contains vertices in X_j^1, which are adjacent in opt* to no vertex in $V_{j'}^\alpha$ with $j' \neq j$, i.e., each vertex in Z_j^0 is adjacent to one bad vertex and one good vertex in Z_j^0. The subset Z_j^1 contains vertices in X_j^1, which are adjacent in opt* to one vertex in $V_{j'}^\alpha$ with $j' \neq j$, i.e., each vertex in Z_j^1 is adjacent to one bad vertex and one good vertex in $Y_{j'} \cup Z_{j'}^1$ for some $j' \neq j$. The following two simple lemmas concerning opt* are useful.

▶ **Lemma 12.** *$\sum_{j=1}^{t_1^{opt}} (2|X_j^2| + |Z_j^0| + |Z_j^1|) = 2t_1^{opt}$.*

Proof. This lemma follows from the definitions of Z_j^0, Z_j^1 and X_j^2. ◀

▶ **Lemma 13.** *For each $j \in [t_1^{opt}]$, $1 \leq |V_j^\alpha \cap V(opt^*)| \leq 2t_1^{opt}$.*

Proof. For each $j \in [t_1^{\text{opt}}]$, exact one of the following three cases applies:

(a) If $Y_j = \emptyset$, then $V_j^\alpha = Z_j^0 \cup Z_j^1 \cup X_j^2 \subseteq V(\text{opt}^*)$ and $1 \leq |V_j^\alpha \cap V(\text{opt}^*)| = |V_j^\alpha| = |Z_j^0| + |Z_j^1| + |X_j^2| \leq 2t_1^{\text{opt}}$.

(b) If $Y_j \neq \emptyset$ and $X_j^1 = Z_j^0 \cup Z_j^1 \neq \emptyset$, then $V_j^\alpha \cap V(\text{opt}^*) = Z_j^0 \cup Z_j^1 \cup X_j^2$ and $1 \leq |V_j^\alpha \cap V(\text{opt}^*)| = |Z_j^0| + |Z_j^1| + |X_j^2| \leq 2t_1^{\text{opt}}$.

(c) If $Y_j \neq \emptyset$ and $X_j^1 = Z_j^0 \cup Z_j^1 = \emptyset$, then $V_j^\alpha \cap V(\text{opt}^*) = X_j^2 \cup \{y_j\}$ and $1 \leq |V_j^\alpha \cap V(\text{opt}^*)| = |X_j^2| + 1 \leq t_1^{\text{opt}} + 1 \leq 2t_1^{\text{opt}}$. ◀

Construction of A^* from opt*. Next we construct a cycle A^* from opt*. Since we shortcut no bad vertex and no good vertex adjacent to bad vertices in opt, $b_s(q_i^{\text{opt}})$ and $b_e(q_i^{\text{opt}})$ for each $i \in [t_1^{\text{opt}}]$ are adjacent in opt* to the same good vertices as in opt, denoted by $g_s(q_i^{\text{opt}})$ and $g_e(q_i^{\text{opt}})$. As defined in Step 2(b) of the case α, for each bad vertex b, $N(b, V_j^\alpha)$ is the set of $\min\{4t_1^{\text{opt}}, |V_j^\alpha|\}$ good vertices in V_j^α, which are closest to b, i.e., $\forall g' \in V_j^\alpha \backslash N(b, V_j^\alpha)$ and $\forall g \in N(b, V_j^\alpha)$, $c(g', b) \geq c(g, b)$. And $N(b) = \bigcup_{j=1}^{t_1^{\text{opt}}} N(b, V_j^\alpha)$. We apply the following procedure to opt*. Initially, set $W^* = \emptyset$. Here, $q_{t_1^{\text{opt}}+1}^{\text{opt}} = q_1^{\text{opt}}$.

From $i = 1$ to $i = t_1^{\text{opt}}$, do the following operations to opt*:

1. Consider V_j^α with $g_e(q_i^{\text{opt}}) \in V_j^\alpha$.
   If $g_e(q_i^{\text{opt}}) \notin N(b_e(q_i^{\text{opt}}), V_j^\alpha)$, then
   (a) Pick an arbitrary vertex denoted by $g_e^*(q_i^{\text{opt}})$ from $N(b_e(q_i^{\text{opt}}), V_j^\alpha) \backslash (W^* \cup V(\text{opt}^*))$.
   (b) Insert $g_e^*(q_i^{\text{opt}})$ between $b_e(q_i^{\text{opt}})$ and $g_e(q_i^{\text{opt}})$ in opt*.
   (c) If $g_e(q_i^{\text{opt}}) \in Z_j^0$, then shortcut $g_e(q_i^{\text{opt}})$.
2. Consider $V_{j'}^\alpha$ with $g_s(q_{i+1}^{\text{opt}}) \in V_{j'}^\alpha$.
   If $g_s(q_{i+1}^{\text{opt}}) \notin N(b_s(q_{i+1}^{\text{opt}}), V_{j'}^\alpha)$, then
   (a) Pick an arbitrary vertex denoted by $g_s^*(q_{i+1}^{\text{opt}})$ from $N(b_s(q_{i+1}^{\text{opt}}), V_{j'}^\alpha) \backslash (W^* \cup V(\text{opt}^*) \cup \{g_e^*(q_i^{\text{opt}})\})$.
   (b) Insert $g_s^*(q_{i+1}^{\text{opt}})$ between $b_s(q_{i+1}^{\text{opt}})$ and $g_s(q_{i+1}^{\text{opt}})$ in opt*.
   (c) If $g_s(q_{i+1}^{\text{opt}}) \in Z_{j'}^0$, then shortcut $g_s(q_{i+1}^{\text{opt}})$.
3. Add $g_e^*(q_i^{\text{opt}})$ and $g_s^*(q_{i+1}^{\text{opt}})$ to W^*.

The above procedure returns a cycle A^* for the following reason. If for some $i \in [t_1^{\text{opt}}]$, we have $g_e(q_i^{\text{opt}}) \in V_j^\alpha$ and $g_e(q_i^{\text{opt}}) \notin N(b_e(q_i^{\text{opt}}), V_j^\alpha)$, then $|N(b_e(q_i^{\text{opt}}), V_j^\alpha)| \geq 4t_1^{\text{opt}}$ by the definition of $N(b_e(q_i^{\text{opt}}), V_j^\alpha)$. By the fact that $|W^*| \leq 2t_1^{\text{opt}} - 2$ during the procedure and Lemma 13, $N(b_e(q_i^{\text{opt}}), V_j^\alpha) \backslash (W^* \cup V(\text{opt}^*)) \neq \emptyset$. By the same argument, $N(b_s(q_{i+1}^{\text{opt}}), V_{j'}^\alpha) \backslash (W^* \cup V(\text{opt}^*) \cup \{g_e^*(q_i^{\text{opt}})\}) \neq \emptyset$. Thus, the output is a cycle. Observe that $c(b_e(q_i^{\text{opt}}), g_e^*(q_i^{\text{opt}})) \leq c(b_e(q_i^{\text{opt}}), g_e(q_i^{\text{opt}}))$ and $c(b_s(q_{i+1}^{\text{opt}}), g_s^*(q_{i+1}^{\text{opt}})) \leq c(b_s(q_{i+1}^{\text{opt}}), g_s(q_{i+1}^{\text{opt}}))$.

Let $H^* = (h_1^*, \dots, h_{t_2^*}^*)$ be the ordered collection of all alternating chains in A^*. Let opt$^{H^*}$ be a minimum-cost TSP-solution, where the vertices in $\bigcup_{i=1}^{t_2^*} V(h_i^*)$ occur in the same alternating chains with the same order as specified in H^*. We get the following lemma as a consequence of Lemma 8.

▶ **Lemma 14.** *$c(A^{min}) \leq 3c(opt^{H^*})$, where A^{min} is the output of Algorithm 3.*

Proof. It is easy to observe that all good vertices in A^*, which are direct neighbors of the t_1^{opt} bad chains, are from $N(b_e(q_i^{\text{opt}}))$ and $N(b_s(q_i^{\text{opt}}))$ for $i \in [t_1^{\text{opt}}]$. Thus, there is a case α' in Step 2(c) where we get an order of alternating chains equal to H^* and by Lemma 8, we have in this case a solution $A^{\alpha'}$ with $c(A^{\alpha'}) \leq 3c(\text{opt}^{H^*})$. Finally, $c(A^{\min}) \leq c(A^{\alpha'}) \leq 3c(\text{opt}^{H^*})$. ◀

Before moving to the third step, we give an upper bound on $c(A^*)$. Let $E_{jj'}^{A^*}$ be the set of edges in A^* connecting a good vertex in V_j^α and a good vertex in $V_{j'}^\alpha$ for $j, j' \in [t_1^{\text{opt}}]$. Then $c(A^*) = \sum_{i=1}^{t_2^*} c(h_i^*) + \sum_{j,j':1\leq j<j'\leq t_1^{\text{opt}}} c(E_{jj'}^{A^*}) + \sum_{j=1}^{t_1^{\text{opt}}} c(E_{jj}^{A^*})$. According to the construction of A^* from opt*, the following three lemmas are easy to see, which provide the foundation of upper-bounding $c(A^*)$.

▶ **Lemma 15.** *$\sum_{i=1}^{t_2^*} c(h_i^*) \leq \sum_{i=1}^{t_2^{opt}} c(h_i^{opt})$, where $H^{opt} = (h_1^{opt}, \dots, h_{t_2^{opt}}^{opt})$ is the ordered collection of all alternating chains in opt.*

Proof. For each $i \in [t_1^{\mathrm{opt}}]$, $(b_e(q_i^{\mathrm{opt}}), g_e(q_i^{\mathrm{opt}})) \in E(\mathrm{opt})$, and exact one of the following two cases applies:
(a) $b_e(q_i^{\mathrm{opt}})$ is adjacent to $g_e(q_i^{\mathrm{opt}})$ in A^*.
(b) $b_e(q_i^{\mathrm{opt}})$ is adjacent to $g_e^*(q_i^{\mathrm{opt}})$ in A^*, and $c(b_e(q_i^{\mathrm{opt}}), g_e^*(q_i^{\mathrm{opt}})) \le c((b_e(q_i^{\mathrm{opt}}), g_e(q_i^{\mathrm{opt}}))$.
This property also holds for the good vertex in A^* adjacent to $b_s(q_i^{\mathrm{opt}})$, for each $i \in [t_1^{\mathrm{opt}}]$. At the same time, A^* contains the same bad chains as opt. Thus, $\sum_{i=1}^{t_2^*} c(h_i^*) \le \sum_{i=1}^{t_2^{\mathrm{opt}}} c(h_i^{\mathrm{opt}})$. ◀

▶ **Lemma 16.** *For all $j' \neq j$ and $j, j' \in [t_1^{opt}]$, $E_{jj'}^{A^*} = E_{jj'}^{opt^*}$, where $E_{jj'}^{opt^*}$ is the set of edges in opt* that connect a vertex in V_j^α and a vertex in $V_{j'}^\alpha$.*

Proof. For each $i \in [t_1^{\mathrm{opt}}]$, there exists some $j \in [t_1^{\mathrm{opt}}]$ such that $g_e(q_i^{\mathrm{opt}}) \in V_j^\alpha$ and $g_e^*(q_i^{\mathrm{opt}}) \in V_j^\alpha$. Inserting $g_e^*(q_i^{\mathrm{opt}})$ between $b_e(q_i^{\mathrm{opt}})$ and $g_e(q_i^{\mathrm{opt}})$ does not add or remove an edge between a good vertex in $V_{j''}^\alpha$ and a good vertex $V_{j'''}^\alpha$ with $j'' \neq j'''$. This property also holds for inserting $g_s^*(q_i^{\mathrm{opt}})$ between $b_s(q_i^{\mathrm{opt}})$ and $g_s(q_i^{\mathrm{opt}})$, for each $i \in [t_1^{\mathrm{opt}}]$.

For each edge $(g, g') \in E_{jj'}^{\mathrm{opt}^*}$ $(j' \neq j)$ where $g \in V_j^\alpha$ and $g' \in V_{j'}^\alpha$, $g \notin Z_j^0$ and $g' \notin Z_{j'}^0$. Then, according to the construction of A^* from opt*, we do not shortcut g or g'. ◀

▶ **Lemma 17.** *For each $j \in [t_1^{opt}]$, $|E_{jj}^{A^*}| \le 2|X_j^2| + |Z_j^0| + |Z_j^1| \le 2t_1^{opt}$.*

Proof. According to the construction of A^* from opt*, it is easy to see that $|E_{jj}^{A^*}| \le 2|X_j^2| + |Z_j^0| + |Z_j^1|$. By Lemma 12, $2|X_j^2| + |Z_j^0| + |Z_j^1| \le 2t_1^{\mathrm{opt}}$, and thus, $|E_{jj}^{A^*}| \le 2t_1^{\mathrm{opt}}$. ◀

▶ **Lemma 18.** *$c(A^*) \le (2k_2 + 1)c(opt)$.*

Proof. Recall that $\mathcal{F}^\alpha = (\mathcal{T}_1^\alpha, \ldots, \mathcal{T}_{t_1^{\mathrm{opt}}}^\alpha)$ is the minimum-cost spanning t_1^{opt}-forest of the graph $G[V^g]$, and $V_j^\alpha = V(\mathcal{T}_j^\alpha)$ for each $j \in [t_1^{\mathrm{opt}}]$. Thus, for each $(g, g') \in E_{jj}^{A^*}$, it trivially holds $c(g, g') \le c(\mathcal{T}_j^\alpha)$. Since opt* has the order H^{opt}, $c(\mathrm{opt}^*) \ge \sum_{i=1}^{t_2^{\mathrm{opt}}} c(h_i^{\mathrm{opt}}) + \sum_{j,j':1\le j<j'\le t_1^{\mathrm{opt}}} c(E_{jj'}^{\mathrm{opt}^*})$. By Lemmas 11, 15, 16, and 17, we have

$$\begin{aligned}
c(A^*) &= \sum_{i=1}^{t_2^*} c(h_i^*) + \sum_{j,j':1\le j<j'\le t_1^{\mathrm{opt}}} c(E_{jj'}^{A^*}) + \sum_{j=1}^{t_1^{\mathrm{opt}}} c(E_{jj}^{A^*}) \\
&\le \sum_{i=1}^{t_2^{\mathrm{opt}}} c(h_i^{\mathrm{opt}}) + \sum_{j,j':1\le j<j'\le t_1^{\mathrm{opt}}} c(E_{jj'}^{\mathrm{opt}^*}) + \sum_{j=1}^{t_1^{\mathrm{opt}}} (|E_{jj}^{A^*}| \cdot c(\mathcal{T}_j^\alpha)) \\
&\le c(\mathrm{opt}^*) + 2t_1^{\mathrm{opt}} \sum_{j=1}^{t_1^{\mathrm{opt}}} c(\mathcal{T}_j^\alpha) \\
&= c(\mathrm{opt}^*) + 2t_1^{\mathrm{opt}} \cdot c(\mathcal{F}^\alpha) \\
&\le c(\mathrm{opt}) + 2k_2 \cdot c(\mathrm{opt}) \\
&= (2k_2 + 1)c(\mathrm{opt}).
\end{aligned}$$
◀

Construction of $\tilde{A}$ from A^*. Finally, we insert all remaining vertices in $V_j^\alpha \backslash V(A^*)$ into A^* for all $j \in [t_1^{\mathrm{opt}}]$ to get a TSP-solution $\tilde{A}$. By the analysis of cases, we obtain the following lemma, which is useful for the choice of locations of insertions.

▶ **Lemma 19.** *For each $j \in [t_1^{opt}]$, at least one of the following three cases applies:*
(a) *$V_j^\alpha \subseteq V(A^*)$.*
(b) *$E_{jj}^{A^*} \neq \emptyset$.*
(c) *There exists $j' \neq j$ such that $E_{jj'}^{A^*} \neq \emptyset$.*

Proof. For each $j \in [t_1^{\text{opt}}]$, at least one of the following four cases applies:

(a) If $Y_j = \emptyset$, then $V_j^\alpha = Z_j^0 \cup Z_j^1 \cup X_j^2 \subseteq V(\text{opt}^*)$ and $|V_j^\alpha| = |Z_j^0| + |Z_j^1| + |X_j^2| \leq 2t_1^{\text{opt}} < 4t_1^{\text{opt}}$. For each $i \in [t_1^{\text{opt}}]$, $N(b_e(q_i^{\text{opt}}), V_j^\alpha) = V_j^\alpha$ and $N(b_s(q_i^{\text{opt}}), V_j^\alpha) = V_j^\alpha$. According to the construction of A^* from opt*, we insert or shortcut no vertex in V_j^α. Thus, $V_j^\alpha \subseteq V(A^*)$.

(b) If $Y_j \neq \emptyset$ and $Z_j^0 \neq \emptyset$, then $E_{jj}^{\text{opt}^*} \neq \emptyset$. According to the construction of A^* from opt*, $E_{jj}^{A^*} \neq \emptyset$.

(c) If $Y_j \neq \emptyset$ and $Z_j^1 \neq \emptyset$, then there exists $j' \neq j$ such that $E_{jj'}^{\text{opt}^*} \neq \emptyset$. By Lemma 16, $E_{jj'}^{A^*} = E_{jj'}^{\text{opt}^*} \neq \emptyset$.

(d) If $Y_j \neq \emptyset$ and $X_j^1 = Z_j^0 \cup Z_j^1 = \emptyset$, then $Y_j \cap V(\text{opt}^*) = \{y_j\}$. There exists $j' \neq j$ such that $E_{jj'}^{\text{opt}^*} \neq \emptyset$. By Lemma 16, $E_{jj'}^{A^*} = E_{jj'}^{\text{opt}^*} \neq \emptyset$. ◀

We insert the vertices in $V_j^\alpha \backslash V(A^*)$ as follows. We double the edges in $\mathcal{T}_j^\alpha$ and shortcut repeated vertices on the $\mathcal{T}_j^\alpha$'s Euler tour, resulting in a cycle C_j^α, where $V(C_j^\alpha) = V_j^\alpha$ and $c(C_j^\alpha) \leq 2c(\mathcal{T}_j^\alpha)$ for each $j \in [t_1^{\text{opt}}]$. We insert the remaining vertices in $V_j^\alpha \backslash V(A^*)$ into A^* according to their orders in C_j^α.

From $j = 1$ to $j = t_1^{\text{opt}}$, do the following operation to A^*:

1. If $V_j^\alpha \subseteq V(A^*)$, then go to the next iteration.
2. If $E_{jj}^{A^*} \neq \emptyset$, then let (g_{j_s}, g_{j_e}) be an arbitrary edge in $E_{jj}^{A^*}$. Traverse the cycle C_j^α, starting from g_{j_s} and shortcutting all vertices not in $(V_j^\alpha \backslash V(A^*)) \cup \{g_{j_s}\}$. Let $C_j' = (g_{j_s}, g_{j_1}, \ldots, g_{j_{l_j}}, g_{j_s})$ be the resulting cycle. Insert the path $(g_{j_1}, \ldots, g_{j_{l_j}})$ between g_{j_s} and g_{j_e} in A^* and remove (g_{j_s}, g_{j_e}). Go to the next iteration.
3. If there is a $j' \neq j$ with $E_{jj'}^{A^*} \neq \emptyset$, then let $(g_{j_s}, g_{j'_e})$ be an arbitrary edge in $E_{jj'}^{A^*}$, where $g_{j_s} \in V_j^\alpha$ and $g_{j'_e} \in V_{j'}^\alpha$. Traverse the cycle C_j^α, starting from g_{j_s} and shortcutting all vertices not in $(V_j^\alpha \backslash V(A^*)) \bigcup \{g_{j_s}\}$. Let $C_j' = (g_{j_s}, g_{j_1}, \ldots, g_{j_{l_j}}, g_{j_s})$ be the resulting cycle.
   (a) If $V_{j'}^\alpha \subseteq V(A^*)$, then insert the path $(g_{j_1}, \ldots, g_{j_{l_j}})$ between g_{j_s} and $g_{j'_e}$ in A^* and remove $(g_{j_s}, g_{j'_e})$.
   (b) If $V_{j'}^\alpha \backslash V(A^*) \neq \emptyset$, then traverse the cycle $C_{j'}^\alpha$, starting from $g_{j'_e}$ and shortcutting all vertices not in $(V_{j'}^\alpha \backslash V(A^*)) \bigcup \{g_{j'_e}\}$. Let $C_{j'}' = (g_{j'_e}, g_{j'_1}, \ldots, g_{j'_{l_{j'}}}, g_{j'_e})$ be the resulting cycle. Then insert the path $(g_{j_1}, \ldots, g_{j_{l_j}}, g_{j'_1}, \ldots, g_{j'_{l_{j'}}})$ between g_{j_s} and $g_{j'_e}$ in A^* and remove $(g_{j_s}, g_{j'_e})$.

After the above iteration, we obtain a new cycle $\tilde{A}$. By Lemma 19, $\tilde{A}$ is a TSP-solution. For each $j \in [t_1^{\text{opt}}]$, since $c(C_j^\alpha) \leq 2c(\mathcal{T}_j^\alpha)$, there is only a small cost increase caused by the insertion of all remaining vertices in $V_j^\alpha \backslash V(A^*)$.

▶ **Lemma 20.** *$c(\tilde{A}) \leq (2k_2 + 3)c(opt)$.*

Proof. We use Δc_j to denote the cost increase caused by the insertion of the vertices in $V_j^\alpha \backslash V(A^*)$ and give an upper bound on Δc_j. In the first case, Δc_j is clearly 0. In the second case, we insert the path $(g_{j_1}, \ldots, g_{j_{l_j}})$ between g_{j_s} and g_{j_e} in A^*. Thus, $\Delta c_j = c(C_j') - c(g_{j_{l_j}}, g_{j_s}) - c(g_{j_s}, g_{j_e}) + c(g_{j_{l_j}}, g_{j_e}) \leq c(C_j') \leq c(C_j^\alpha) \leq 2c(\mathcal{T}_j^\alpha)$. Case 3(a) is identical to the second case. In Case 3(b) we add two paths in A^* involving V_j^α and $V_{j'}^\alpha$. By a similar analysis, we have both $\Delta c_j \leq 2c(\mathcal{T}_j^\alpha)$ and $\Delta c_{j'} \leq 2c(\mathcal{T}_{j'}^\alpha)$. By Lemmas 11 and 18, we conclude $c(\tilde{A}) = c(A^*) + \sum_{j=1}^{t_1^{\text{opt}}} \Delta c_j \leq c(A^*) + 2\sum_{j=1}^{t_1^{\text{opt}}} c(\mathcal{T}_j^\alpha) = c(A^*) + 2c(\mathcal{F}^\alpha) \leq (2k_2 + 1)c(\text{opt}) + 2c(\text{opt}) = (2k_2 + 3)c(\text{opt})$. ◀

Now, we have all tools to prove the main result of this subsection.

▶ **Theorem 21.** *Algorithm 3 is a parameterized* $(6k_2+9)$*-approximation algorithm for TSP running in* $O(k_2^{O(k_2)} \cdot n^3)$ *time, where* k_2 *is the minimum number of vertices whose removal results in a metric graph.*

Proof. Recall that H^* is the order of all alternating chains in A^*. From A^* to $\tilde{A}$, we insert the remaining good vertices between two good vertices without changing the good vertices which are direct neighbors of bad chains. Thus, the alternating chains in $\tilde{A}$ occur in the same order in A^*, and then $c(\tilde{A}) \geq c(\mathrm{opt}^{H^*})$. By Lemmas 14 and 20, $c(A^{\min}) \leq 3c(\mathrm{opt}^{H^*}) \leq 3c(\tilde{A}) \leq (6k_2+9)c(\mathrm{opt})$. Combining with Lemma 10, we conclude that Algorithm 3 returns a $(6k_2+9)$-approximation solution in $O(k_2^{O(k_2)} \cdot n^3)$ time. ◀

5 Conclusion

Based on the concept of "distance from approximability", we present two parameterized approximation algorithms for TSP on general graphs, parameterized by the number k_1 of violating triangles or the minimum size k_2 of violating vertex sets, which achieve approximation factors of 3 and $6k_2$+9, respectively. These seem to be the first parameterized approximation algorithms for TSP on general graphs. It remains open whether the factor $6k_2$+9 can be improved to a constant. Moreover, in comparison with the $O(2^n)$-time exact algorithm [4, 21], it is crucial to improve the running time of the approximation algorithms to increase their practical relevance. Finally, it is an interesting research direction to apply the concept of "distance from approximability" to other optimization problems.

References

1 Thomas Andreae. On the traveling salesman problem restricted to inputs satisfying a relaxed triangle inequality. *Networks*, 38(2):59–67, 2001.

2 Thomas Andreae and Hans-Jürgen Bandelt. Performance guarantees for approximation algorithms depending on parametrized triangle inequalities. *SIAM Journal on Discrete Mathematics*, 8(1):1–16, 1995.

3 Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

4 Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM*, 9(1):61–63, 1962.

5 Michael A. Bender and Chandra Chekuri. Performance guarantees for the TSP with a parameterized triangle inequality. *Information Processing Letters*, 73(1-2):17–21, 2000.

6 Hans-Joachim Böckenhauer, Juraj Hromkovič, Ralf Klasing, Sebastian Seibert, and Walter Unger. Towards the notion of stability of approximation for hard optimization tasks and the traveling salesman problem. *Theoretical Computer Science*, 285(1):3–24, 2002.

7 Hans-Joachim Bockenhauer, Juraj Hromkovic, Joachim Kneis, and Joachim Kupke. The parameterized approximability of TSP with deadlines. *Theory of Computing Systems*, 41(3):431–444, 2007.

8 Edouard Bonnet, Bruno Escoffier, Eun Jung Kim, and Vangelis Th. Paschos. On subexponential and FPT-time inapproximability. *Algorithmica*, 71(3):541–565, 2015.

9 Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From Gap-ETH to FPT-inapproximability: Clique, dominating set, and more. In *Proceedings of the IEEE 58th Annual Symposium on Foundations of Computer Science*, pages 743–754. IEEE, 2017.

10 Yijia Chen and Bingkai Lin. The constant inapproximability of the parameterized dominating set problem. In *Proceedings of the IEEE 57th Annual Symposium on Foundations of Computer Science*, pages 505–514. IEEE, 2016.

11 Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Carnegie-Mellon University, 1976.

12 Vladimir G. Deĭneko, Michael Hoffmann, Yoshio Okamoto, and Gerhard J. Woeginger. The traveling salesman problem with few inner points. *Operations Research Letters*, 34(1):106–110, 2006.

13 Erik D. Demaine and MohammadTaghi Hajiaghayi. Equivalence of local treewidth and linear local treewidth and its algorithmic applications. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 840–849. SIAM, 2004.

14 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

15 Andreas Emil Feldmann, C.S. Karthik, Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020.

16 Michael R. Garey and David S. Johnson. Computers and intractability: A guide to the theory of NP-completeness. *Bulletin (New Series) of the AMS*, 3(2):898–904, 1980.

17 Lee-Ad Gottlieb. A light metric spanner. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 759–772. IEEE, 2015.

18 Martin Grohe. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica*, 23:613–632, 2003.

19 Martin Grohe, Ken-ichi Kawarabayashi, and Bruce Reed. A simple algorithm for the graph minor decomposition - logic meets structural graph theory-. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 414–431. SIAM, 2013.

20 Jiong Guo, Falk Hüffner, and Rolf Niedermeier. A structural view on parameterizing problems: Distance from triviality. In *Proceedings of the 1st International Workshop on Parameterized and Exact Computation*, pages 162–173. Springer, 2004.

21 Michael Held and Richard M. Karp. A dynamic programming approach to sequencing problems. *Journal of the SIAM*, 10(1):196–210, 1962.

22 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 32–45. ACM, 2021.

23 Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

24 Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Structural parameters, tight bounds, and approximation for (k,r)-center. *Discrete Applied Mathematics*, 264:90–117, 2019.

25 Michael Khachay and Katherine Neznakhina. Approximability of the minimum-weight k-size cycle cover problem. *Journal of Global Optimization*, 66(1):65–82, 2016.

26 Christian Knauer and Andreas Spillner. A fixed-parameter algorithm for the minimum weight triangulation problem based on small graph separators. In *Proceedings of the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 49–57. Springer, 2006.

27 Bingkai Lin. Constant approximating k-clique is W[1]-hard. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1749–1756. ACM, 2021.

28 Dániel Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

29 Usha Mohan, Sivaramakrishnan Ramani, and Sounaka Mishra. Constant factor approximation algorithm for TSP satisfying a biased triangle inequality. *Theoretical Computer Science*, 657:111–126, 2017.

30 Tobias Mömke. An improved approximation algorithm for the traveling salesman problem with relaxed triangle inequality. *Information Processing Letters*, 115(11):866–871, 2015.

31 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

32 Christos H. Papadimitriou. The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237–244, 1977.

33 Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1283–1296. ACM, 2018.

34 Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.

35 Anatolii Ivanovich Serdyukov. On some extremal walks in graphs. *Upravliaemie systemy*, 17:76–79, 1978.