FPT Approximation Algorithms for TSP on Non-Metric Graphs

Jingyang Zhao[1,2], **Zimo Sheng**[3], **Mingyu Xiao**[1*]

[1]University of Electronic Science and Technology of China
[2]Kyung Hee University, Yongin-si, South Korea
[3]Anhui University
jingyangzhao1020@gmail.com, shengzimo2016@gmail.com, myxiao@uestc.edu.cn

Abstract

TSP is a classic and extensively studied problem with numerous real-world applications in artificial intelligence and operations research. It is well-known that TSP admits a constant approximation ratio on metric graphs but becomes NP-hard to approximate within any computable function $f(n)$ on general graphs. This disparity highlights a significant gap between the results on metric graphs and general graphs. Recent research has introduced some parameters to measure the "distance" of general graphs from being metric and explored FPT approximation algorithms parameterized by these parameters. Two commonly studied parameters are p, the number of vertices in triangles violating the triangle inequality, and q, the minimum number of vertices whose removal results in a metric graph. In this paper, we present improved FPT approximation algorithms with respect to these two parameters. For p, we propose an FPT algorithm with a 1.5-approximation ratio, improving upon the previous ratio of 2.5. For q, we significantly enhance the approximation ratio from 11 to 3, advancing the state of the art in both cases.

Introduction

The Traveling Salesman Problem (TSP) is a well-known optimization problem with numerous real-world applications (Applegate 2006), including logistics, genetics, manufacturing, and telecommunications. Given an edge-weighted complete graph $G = (V, E, w)$ with n vertices and a non-negative edge weight function $w : E \to \mathbb{R}_{\geq 0}$, the objective is to find a minimum-weight TSP tour (i.e., Hamiltonian cycle) that visits each vertex exactly once. Due to its NP-hardness (Garey and Johnson 1979), TSP and its variants have been extensively studied in the context of approximation algorithms (Saller, Koehler, and Karrenbauer 2025; Traub and Vygen 2025).

While TSP on general graphs is NP-hard to approximate within any computable function $f(n)$ (Sahni and Gonzalez 1976), its widely-used variant on metric graphs (metric TSP), where the weight function satisfies the triangle inequality, admits a constant approximation ratio. Notably, Christofides (2022) and Serdyukov (1978) independently proposed the famous 1.5-approximation algorithm for metric TSP. Later, Karlin *et al.* (2021; 2023) improved the ratio to $1.5 - 10^{-36}$.

Given the significant gap between the results on general graphs and metric graphs, many approximation algorithms have been developed for graphs that are "nearly" metric, aiming to measure the "distance" from approximability. In the literature, two main research directions are explored. The first direction relaxes the condition for the triangle inequality to hold, for instance, by satisfying the *τ-triangle inequality*: $\tau \cdot (w(a, b) + w(b, c)) \geq w(a, c)$ for all $a, b, c \in V$. The second direction relaxes the scope of the triangle inequality, where it is only required to hold for a subset of the vertices.

For the first direction, a comprehensive research framework has been established (see the survey (Klasing and Mömke 2018)). Andreae and Bandelt (1995) gave a $(3\tau^2 + \tau)/2$-approximation algorithm for TSP. Bender and Chekuri (2000) improved this ratio to 4τ when $\tau > 7/3$, and Mömke (2015) further improved the ratio to $3(\tau^2 + \tau)/4$ for small τ. More improvements can be found in (Andreae 2001; Böckenhauer et al. 2002).

For the second direction, two lines of research have made explorations. First, for a graph where the vertex set can be partitioned into two subsets V_1 and V_2 such that both $G[V_1]$ and $G[V_2]$ are metric, Mohan (2017) proposed a 3.5-approximation algorithm. Further, when the vertex set is partitioned into k groups and the tour must visit vertices within each group consecutively, the problem is known as *Subgroup Path Planning*. In this problem, the triangle inequality $w(a, b) + w(b, c) \geq w(a, c)$ may be violated only when vertices a and c belong to the same group while b belongs to a different one. This problem has applications in AI, such as for polishing robots (Safilian et al. 2016). Sumita *et al.* (2017) first proposed a 3-approximation algorithm, which was recently improved to 2.167 (Zhao et al. 2025). Moreover, when each group has a size of at most 2, the problem reduces to *Subpath Planning* (Safilian et al. 2016), which has applications in electronic printing and admits a 1.5-approximation ratio (Sumita et al. 2017).

In fact, a natural direction for relaxing the scope of the triangle inequality is to consider graphs where only a small number of vertices are involved in triangles that violate the inequality. A practical example is urban tour bus routing. Operators design TSP tours to visit landmarks, but exclusive

*Corresponding author

express routes between certain high-profile landmarks create shortcuts that violate the triangle inequality. These violations are typically caused by a limited number of landmarks, meaning the metric property could be restored by removing just a few critical nodes.

Recently, this direction has been investigated using techniques from parameterized complexity. For a minimization problem, an algorithm is called an Fixed-Parameter Tractable (FPT) ρ-approximation algorithm if, for any instance $\mathcal{I}$ with a parameter k, it runs in $\mathcal{O}(f(k)) \cdot poly(|\mathcal{I}|)$ time and returns a ρ-approximate solution, where $f(k)$ is a computable function of k. Such algorithms are particularly useful when k is small (Cygan et al. 2015). Following this line, Zhou *et al.* (2022; 2025) introduced two natural parameters: the number of triangles p' that violate the triangle inequality, and the minimum number of vertices q whose removal results in a metric graph. For TSP parameterized by p', they presented an FPT 3-approximation algorithm, which was recently improved to 2.5 by Bampis *et al.* (2024). For TSP parameterized by q, they gave an FPT 11-approximation algorithm.

Our research follows this line to study FPT approximation algorithms for TSP with respect to the parameters p and q, where p denotes the number of vertices involved in triangles that violate the triangle inequality. Although $p' = \Omega(p)$, existing FPT approximation algorithms for TSP with respect to p' also extend directly to p, as noted by Bampis *et al.* (2024). Moreover, FPT approximation algorithms with respect to the parameters p and q have also been studied for several location problems (Zhou et al. 2025).

Our results

In this paper, we present improved FPT approximation algorithms for TSP with respect to the parameters p and q, addressing the question raised in (Bampis, Escoffier, and Xefteris 2024) regarding the potential for further improvement in the approximation ratio. Let α denote the approximation ratio for metric TSP. Now, $\alpha = 1.5 - 10^{-36}$ (Karlin, Klein, and Gharan 2023). Our results are summarized as follows.

For the parameter p, we first introduce a simple $(\alpha+1)$-approximation algorithm with runtime $2^{\mathcal{O}(p)} + n^{\mathcal{O}(1)}$. This improves the previous 2.5-approximation algorithm with runtime $2^{\mathcal{O}(p \log p)} \cdot n^{\mathcal{O}(1)}$ (Bampis, Escoffier, and Xefteris 2024). Then, we propose a more refined 1.5-approximation algorithm with runtime $2^{\mathcal{O}(p \log p)} \cdot n^{\mathcal{O}(1)}$.

For the parameter q, we propose an FPT 3-approximation algorithm, which significantly improves the previous FPT 11-approximation algorithm (Zhou, Zhang, and Guo 2025). Both algorithms run in $2^{\mathcal{O}(q \log q)} \cdot n^{\mathcal{O}(1)}$ time. Our result also matches the approximation ratio of an earlier algorithm with a non-FPT runtime of $n^{\mathcal{O}(q+1)}$ (Zhou, Li, and Guo 2022).

A summary of the previous and our results is provided in Table 1. Our improvements arise from the application of novel techniques for handling non-metric graphs, which may also prove valuable for tackling other related problems.

Due to limited space, the proofs of lemmas and theorems marked with "*" were omitted, and they can be found in the full version of this paper (Zhao, Sheng, and Xiao 2025).

Param.	Approx.	Runtime	Reference
p	3	$2^{\mathcal{O}(p \log p)} \cdot n^{\mathcal{O}(1)}$	Zhou *et al.* (2022)
	2.5	$2^{\mathcal{O}(p \log p)} \cdot n^{\mathcal{O}(1)}$	Bampis *et al.* (2024)
	$\alpha+1$	$2^{\mathcal{O}(p)} + n^{\mathcal{O}(1)}$	**This paper**
	1.5	$2^{\mathcal{O}(p \log p)} \cdot n^{\mathcal{O}(1)}$	
q	3	$n^{\mathcal{O}(q+1)}$	Zhou *et al.* (2022)
	11	$2^{\mathcal{O}(q \log q)} \cdot n^{\mathcal{O}(1)}$	Zhou *et al.* (2025)
	3	$2^{\mathcal{O}(q \log q)} \cdot n^{\mathcal{O}(1)}$	**This paper**

Table 1: A summary of the previous and our results on approximation algorithms for TSP parameterized by p and q.

Preliminary

Let $G = (V, E, w)$ be a complete input graph with $|V| = n$ and edge weights $w : E \to \mathbb{R}_{>0}$. For any subset $E' \subseteq E$, define the total weight as $w(E') \coloneqq \sum_{e \in E'} w(e)$. We say that G is a *metric* graph if, for all $a, b, c \in V$, the following conditions hold: $w(a, a) = 0$, $w(a, b) = w(b, a)$, and the triangle inequality $w(a, b) \leq w(a, c) + w(c, b)$.

Although the input graph contains no multiple edges, some operations (e.g., union of two graphs) may introduce multi-edges. Hence, we allow multi-edge graphs throughout the paper. For any $V' \subseteq V$, let $G[V']$ denote the subgraph of G induced by V'. For any subgraph $G' \subseteq G$, let $V(G')$ and $E(G')$ denote its vertex and edge sets, respectively.

A *spanning t-forest* $\mathcal{F}$ in G is a set of t vertex-disjoint trees that together cover all vertices in V. A *matching* $\mathcal{M}$ in G is a set of vertex-disjoint edges such that $V(\mathcal{M}) = V$.

A triangle on three distinct vertices a, b, and c is denoted by $\Delta(a, b, c)$. We say that a triangle is *violating* if it violates the triangle inequality. A subset $V' \subseteq V$ is called a *violating set* (VS) if removing V' from G results in a metric subgraph $G[V \setminus V']$. Let p be the number of vertices involved in violating triangles, and q the size of a minimum VS.

For any (multi-)graph $G' = (V', E', w')$, the *degree* of a vertex is the number of incident edges. Let $Odd(G')$ denote the set of odd-degree vertices. Then, G' is *Eulerian* if it is connected and $Odd(G') = \emptyset$. A *tour* $T = \overline{v_1 v_2 \dots v_s}$ in G' is a sequence of vertices (allowing repetitions), inducing the edge set $E'(T) \coloneqq \{v_1v_2, v_2v_3, \dots, v_sv_1\}$, and weight $w'(T) \coloneqq w'(E'(T))$. Moreover,

- T is a *simple tour* if it has no repeated vertices, and a *TSP tour* if it further satisfies $|E'(T)| = |V'|$;
- T is an *Eulerian tour* if $E'(T)$ is exactly the set of all edges in G'. We may use G' to refer to its Eulerian tour.

Given a tour $T = \overline{...xyz...}$, removing y yields a new tour $\overline{...xz...}$, called *shortcutting* the vertex y (or edges xy and yz), or *taking a shortcut* on T. A simple tour $\overline{v_1 v_2 ... v_s}$ in G' defines an *s-path* $A = v_1v_2...v_s$, consisting of $s-1$ edges in $E'(A) \coloneqq \{v_1v_2, v_2v_3, ..., v_{s-1}v_s\}$. A 1-path is a single vertex, and a $|V'|$-path in G' is called a *TSP path*. A path is also referred to as a *chain*.

Throughout the paper, let T^* denote an optimal TSP tour in the graph G, and let OPT denote its weight.

In our algorithms, the term "guess" refers to enumerating all possible candidates and selecting the best found solution.

Algorithm 1: ALG.1

Input: An instance $G = (V = V_g \cup V_b, E, w)$.
Output: A feasible solution.

1: Arbitrarily select a good vertex $o \in V_g$.
2: Compute an optimal TSP tour T_b in $G[V_b \cup \{o\}]$ using the DP method (Bellman 1962).
3: Compute an α-approximate TSP tour T_g in $G[V_g]$ by applying an α-approximation algorithm for metric TSP.
4: Take a shortcut on $T_b \cup T_g$ to obtain a TSP tour T_1 in G.

For convenience, we may regard a set of edges as a graph and a set of trees as the union of their edge sets. We interpret such combinations as the graph formed by all involved edges.

TSP Parameterized by p

We first define some notations. A vertex is *bad* if it appears in a violating triangle, and *good* otherwise. Let $V = V_b \cup V_g$, where V_b (resp., V_g) denotes the set of bad (resp., good) vertices. Note that $|V_b| = p$. We assume $\min\{|V_g|, |V_b|\} \geq 3$, as otherwise one can easily obtain an FPT α-approximation algorithm with runtime $2^{\mathcal{O}(p)} + n^{\mathcal{O}(1)}$ (details are provided in the full version).

By definition, we have the following property.

Property 1. *Every triangle containing a good vertex satisfies the triangle inequality.*

The first algorithm

In this subsection, we give an FPT $(\alpha + 1)$-approximation algorithm (ALG.1) with runtime $2^{\mathcal{O}(p)} + n^{\mathcal{O}(1)}$.

Our ALG.1 is simple. First, it arbitrarily selects a good vertex o. Then, it computes an optimal TSP tour T_b in $G[V_b \cup \{o\}]$ using the DP method (Bellman 1962; Held and Karp 1962). Next, it computes an α-approximate TSP tour T_g in $G[V_g]$ by applying an α-approximation algorithm for metric TSP. Finally, it constructs a TSP tour T_1 in G by taking a shortcut on $T_b \cup T_g$.

The TSP tour T_b in ALG.1 contains exactly one good vertex. This trick enables us to take a shortcut on $T_b \cup T_g$ to obtain a TSP tour in G without increasing the weight.

Our ALG.1 is described in Algorithm 1.

Lemma 1 (*). *It holds that OPT $\geq w(T_b^*)$ and OPT $\geq w(T_g^*)$, where T_b^* (resp., T_g^*) denotes an optimal TSP tour in $G[V_b \cup \{o\}]$ (resp., $G[V_g]$).*

Theorem 1. *For TSP parameterized by p, ALG.1 achieves an FPT $(\alpha+1)$-approximation with runtime $2^{\mathcal{O}(p)} + n^{\mathcal{O}(1)}$.*

Proof. First, we analyze the solution quality of ALG.1.

Since T_1 is obtained by taking a shortcut on $T_b \cup T_g$, we assume that $T_1 = \overline{ou_1...u_pv_1...v_{n-p-1}}$, where $T_b = \overline{ou_1...u_p}$ and $T_g = \overline{ov_1...v_{n-p-1}}$. Since o is a good vertex, the triangle $\Delta(o, v_1, u_p)$ satisfies the triangle inequality by Property 1, which implies $w(u_p, o) + w(o, v_1) \geq w(u_p, v_1)$. Thus, we have $w(T_b)+w(T_g) \geq w(T_1)$. Since T_b (resp., T_g) is an optimal (resp., α-approximate) TSP tour in $G[V_b \cup \{o\}]$ (resp., $G[V_g]$), by Lemma 1, we have $(\alpha+1)\cdot\text{OPT} \geq w(T_1)$.

Next, we analyze the runtime of ALG.1.

Algorithm 2: ALG.2

Input: An instance $G = (V = V_g \cup V_b, E, w)$.
Output: A feasible solution.

1: Guess the set of bad chains $\mathcal{A}$ in T^*.
2: Construct a complete graph $\widetilde{G} = (V_{\mathcal{A}} \cup V_g, \widetilde{E}, \widetilde{w})$ by contracting each $A \in \mathcal{A}$ into a vertex u_A, where $V_{\mathcal{A}}$ is the set of contracted vertices, and $\widetilde{w}$ is defined as follows: (1) $\widetilde{w}(u_A, v) = \min\{w(u, v), w(u', v)\}$ for any $A = u...u' \in \mathcal{A}$ and $v \in V_g$; (2) $\widetilde{w}(v, v') = w(v, v')$ for any $v, v' \in V_g$; (3) $\widetilde{w}(u_A, u_{A'}) = +\infty$ for any $A, A' \in \mathcal{A}$.
3: Find a minimum spanning tree $\widetilde{F}$ in $\widetilde{G}$, and obtain the corresponding set of edge F in G with $w(F) = \widetilde{w}(\widetilde{F})$.
4: Let $F_{\mathcal{A}} = F \cup \mathcal{A}$, which forms a CST in G.
5: Construct a complete graph $G' = (V' = Odd(F_{\mathcal{A}}), E', w')$: For any $u, u' \in V'$, set $w'(u, u') = w(A)$ if $u...u' = A \in \mathcal{A}$, and $w'(u, u') = w(u, u')$ otherwise.
6: Find a minimum matching $\mathcal{M}'_{\mathcal{A}}$ in G', and obtain the corresponding set of edges $\mathcal{M}_{\mathcal{A}}$ in G with $w(M_{\mathcal{A}}) = w'(M'_{\mathcal{A}})$.
7: Obtain a TSP tour T_2 in G (see Lemma 3).

Steps 1, 3, and 4 take polynomial time, i.e., $n^{\mathcal{O}(1)}$, while Step 2 is dominated by the computation of an optimal TSP tour. Since $|V_b|+1 = p+1$, the optimal TSP tour can be computed in $2^{\mathcal{O}(p)}$ time using the DP method (Bellman 1962). Thus, the overall runtime is $2^{\mathcal{O}(p)} + n^{\mathcal{O}(1)}$. □

By Theorem 1, using the $(1.5 - 10^{-36})$-approximation algorithm (Karlin, Klein, and Gharan 2021) for metric TSP, ALG.1 achieves an FPT approximation ratio of $2.5 - 10^{-36}$, improving the algorithm in (Bampis, Escoffier, and Xefteris 2024) in both approximation ratio and runtime.

The second algorithm

In this subsection, we give an FPT 1.5-approximation algorithm (ALG.2) with runtime $2^{\mathcal{O}(p \log p)} \cdot n^{\mathcal{O}(1)}$.

ALG.2 proceeds as follows. To handle bad vertices, it first guesses the subgraph $T^*[V_b]$, i.e., the subgraph of the optimal TSP tour T^* induced by V_b. This forms a set of paths on bad vertices, called *bad chains* $\mathcal{A}$.

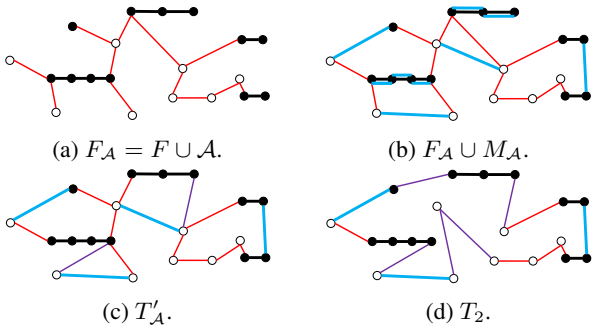
ALG.2 then augments $\mathcal{A}$ into a TSP tour in G. It first constructs a minimum *constrained spanning tree (CST)* $F_{\mathcal{A}}$ in G, i.e., a tree such that $E(\mathcal{A}) \subseteq E(F_{\mathcal{A}})$, each i-degree vertex in $\mathcal{A}$ is incident to exactly i bad vertices in $F_{\mathcal{A}}$, and each 2-degree vertex in $\mathcal{A}$ remains of degree 2 in $F_{\mathcal{A}}$. It then adds a set of edges $\mathcal{M}_{\mathcal{A}}$ to fix the parity of $Odd(F_{\mathcal{A}})$ and computes a TSP tour T_2 in G by shortcutting on $F_{\mathcal{A}} \cup M_{\mathcal{A}}$.

Algorithm 2 gives full details, with an example in Fig. 1.[1]

We first compare our ALG.2 with previous algorithms.

The previous FPT 3-approximation algorithm (Zhou, Li, and Guo 2022) and 2.5-approximation algorithm (Bampis, Escoffier, and Xefteris 2024) follow a similar framework: they guess the set of bad chains $\mathcal{A}$, extract a set V'_b of endpoint vertices from $\mathcal{A}$, and then find a minimum spanning

[1]Strictly speaking, Steps 2-7 are conceptually nested within Step 1, as they depend on the guessed set $\mathcal{A}$ in Step 1. That is, the algorithm enumerates all possible choices of $\mathcal{A}$, and for each choice, it executes Steps 2-7. However, for clarity and conciseness, we present the algorithm in a sequential manner.



(a) $F_{\mathcal{A}} = F \cup \mathcal{A}$. (b) $F_{\mathcal{A}} \cup M_{\mathcal{A}}$. (c) $T'_{\mathcal{A}}$. (d) T_2.

Figure 1: An illustration of our ALG.2, where each black (resp., white) node denotes a bad (resp., good) vertex. In (a), the edges in $E(F)$ (resp., $E(\mathcal{A})$) are shown in red (resp., black); In (b), the edges in $\mathcal{M}_{\mathcal{A}}$ are shown in blue; In (c), $\mathcal{T}'_{\mathcal{A}}$ is obtained by shortcutting repeated edges on the paths in $\mathcal{A} \cap \mathcal{A}'$ (see Lemma 3), with the new edges shown in purple; In (d), the TSP tour T_2 is obtained by taking shortcuts on $\mathcal{T}'_{\mathcal{A}}$ (see Lemma 3), with the new edges also shown in purple.

$|\mathcal{A}|$-forest $\mathcal{F}$ in $G[V'_b \cup V_g]$ (Khachay and Neznakhina 2016), where each vertex $b_i \in V'_b$ lies in a different tree. By adding a set of edges E', they then form a TSP tour on V_b. The algorithm in (Zhou, Li, and Guo 2022) takes shortcuts on $\mathcal{A} \cup E' \cup 2\mathcal{F}$ for a 3-approximation, while the algorithm in (Bampis, Escoffier, and Xefteris 2024) uses a matching $\mathcal{M}$ on $Odd(\mathcal{A} \cup E' \cup \mathcal{F})$ to shortcut $\mathcal{A} \cup E' \cup \mathcal{F} \cup \mathcal{M}$ to achieve a 2.5-approximation.

In contrast, ALG.2 computes a CST $F_{\mathcal{A}}$ in G, avoiding the need for E'. However, $G[Odd(F_{\mathcal{A}})]$ may be non-metric, so we cannot simply use a minimum matching (Lawler 1976). We build an auxiliary graph G' (Step 5) and find a matching $\mathcal{M}'_{\mathcal{A}}$ in G', then shortcut on $F_{\mathcal{A}} \cup M_{\mathcal{A}}$ to obtain a TSP tour, where $\mathcal{M}_{\mathcal{A}}$ is the corresponding set of edges in G.

Hence, ALG.2 introduces key differences and new ideas, requiring refined analysis for the final approximation ratio.

Lemma 2 (*). *ALG.2 computes a CST $F_{\mathcal{A}}$ with $w(F_{\mathcal{A}}) \leq$ OPT, and a set of edges $\mathcal{M}_{\mathcal{A}}$ with $w(M_{\mathcal{A}}) \leq \frac{1}{2} \cdot$ OPT.*

Lemma 3. *In Step 6 of ALG.2, a TSP tour T_2 in G can be found in $\mathcal{O}(n)$ time with weight at most $\frac{3}{2} \cdot$ OPT.*

Proof. Clearly, $F_{\mathcal{A}} \cup M_{\mathcal{A}}$ forms an Eulerian graph $G_{\mathcal{A}}$ with $V(G_{\mathcal{A}}) = V$ and $\mathcal{O}(n)$ edges. Hence, an Eulerian tour $T_{\mathcal{A}}$ can be computed in $\mathcal{O}(n)$ time (Cormen et al. 2022).

Note that $w(T_{\mathcal{A}}) = w(F_{\mathcal{A}}) + w(M_{\mathcal{A}}) \leq \frac{3}{2} \cdot$OPT. To prove the lemma, it suffices to show how to compute in $\mathcal{O}(n)$ time a TSP tour T_2 in G with weight at most $T_{\mathcal{A}}$.

We first observe that

(1) the edges between bad vertices in $E(F_{\mathcal{A}})$ are exactly those in $E(\mathcal{A})$;
(2) the edges between bad vertices in $E(M_{\mathcal{A}})$ form a set of vertex-disjoint paths $\mathcal{A}'$ since $\mathcal{M}'_{\mathcal{A}}$ is a matching in G'.

By Step 5 of ALG.2, we further know that

(3) for each path $A \in \mathcal{A}'$, either $A \in \mathcal{A}$, or A is edge-disjoint from all paths in $\mathcal{A}$.

Thus, the edges on the paths in $\mathcal{A} \cap \mathcal{A}'$ (resp., $\mathcal{A} \setminus \mathcal{A}'$) appear exactly twice (resp., once) in $T_{\mathcal{A}}$ (see (b) in Fig.1).

Next, we shortcut the repeated edges on the paths in $\mathcal{A} \cap \mathcal{A}'$. Consider any path $ab...b'a' \in \mathcal{A} \cap \mathcal{A}'$. By (1), (2), and (3), in V_b, there is only one vertex b (resp., b') adjacent to a (resp., a'). Since $G_{\mathcal{A}}$ is also Eulerian, $G_{\mathcal{A}}$ is connected and both a and a' have even degree. So, either a or a', say a, must be adjacent to at least two good vertices. Thus, we can use a good neighbor of a to shortcut the repeated edges on the path $ab...b'a'$ without increasing the weight by Property 1.

Denote the new Eulerian tour by $T'_{\mathcal{A}}$. We know that

(4) each edge between bad vertices appears once, and each bad vertex is incident to at most two bad vertices (see (c) in Fig.1).

Then, we shortcut the repeated bad vertices in $T'_{\mathcal{A}}$. Suppose a bad vertex a appears at least twice in $T'_{\mathcal{A}}$. By (4), each appearance of a, except for one, must be adjacent to at least one good vertex. Thus, each of these can be shortcut via a good neighbor without increasing the weight by Property 1.

Last, repeated good vertices can clearly be shortcut without increasing the weight. So, given $\mathcal{T}_{\mathcal{A}}$, a TSP tour T_2 in G can be found with weight at most $w(\mathcal{T}_{\mathcal{A}})$ in $\mathcal{O}(n)$ time. □

Theorem 2 (*). *For TSP parameterized by p, ALG.2 has an FPT 1.5-approximation ratio with runtime $2^{\mathcal{O}(p \log p)} \cdot n^{\mathcal{O}(1)}$.*

TSP Parameterized by q

In this section, we consider the parameter q. It is NP-hard to compute q. In our algorithm, we first compute a minimum violating set (VS) V_b in $\mathcal{O}(3^q n^3)$ time by using the algorithm in (Zhou, Li, and Guo 2022). Let $V_g = V \setminus V_b$. Vertices in V_b are called *bad*, and vertices in V_g are called *good*. Now $|V_b| = q$. We assume that $\min\{|V_g|, |V_b|\} \geq 1$. Next, we first define some notations.

In T^*, edges between bad vertices form the set of *bad chains* $\mathcal{A}$; edges between a bad and a good vertex form the set $\mathcal{B}$, where each such edge is a *limb*; and edges between good vertices form the set of *good chains* $\mathcal{R}$. Note that

$$\text{OPT} = w(\mathcal{A}) + w(\mathcal{B}) + w(\mathcal{R}) \text{ and } |\mathcal{A}| = 2|\mathcal{B}| = |\mathcal{R}|. \quad (1)$$

Let V_g^i denote the set of vertices in V_g that are incident to i bad vertices in T^*, where $i \in \{0, 1, 2\}$. The vertices in V_g^0 (resp., $V_g^1 \cup V_g^2$) are called *internal vertices* (resp., *anchors*). An anchor is a *single anchor* if it is in V_g^2, and a *pair anchor* otherwise. Let $V_a = V_g^1 \cup V_g^2$ be the set of all anchors. Then,

$$|\mathcal{A}| = \frac{1}{2}|\mathcal{B}| = |\mathcal{R}| \leq q \quad \text{and} \quad |V_a| \leq 2q. \quad (2)$$

An illustration of these notations can be found in Fig.2.

The algorithm

In this subsection, we propose an FPT 3-approximation algorithm (ALG.3) with runtime $2^{\mathcal{O}(q \log q)} \cdot n^{\mathcal{O}(1)}$.

The high-level idea of ALG.3 is as follows. Since a triangle containing one bad vertex and two good vertices may violate the triangle inequality, Property 1 in the previous

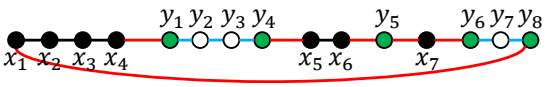


Figure 2: An illustration of our notations: the cycle denotes T^*; black, white, and green nodes denote bad vertices, internal vertices, and anchors, respectively. The sets are as follows: $\mathcal{A} = \{x_1...x_4, x_5x_6, x_7\}$, $\mathcal{B} = \{x_4y_1, ..., y_8x_1\}$, $\mathcal{R} = \{y_1...y_4, y_5, y_6y_7y_8\}$. Note that $x_7 \in \mathcal{A}$ is a 1-path.

Algorithm 3: ALG.3

Input: An instance $G = (V = V_g \cup V_b, E, w)$.
Output: A feasible solution.

1: Guess the set of bad chains $\mathcal{A}$ in T^*.
2: Find a minimum spanning $|\mathcal{A}|$-forest $\mathcal{F}$ in $G[V_g]$.
3: Guess a set of limbs $\mathcal{B}'$ by calling LIMB.
4: Guess a set of edges $\mathcal{R}'$ by calling CONNECT.
5: Find a set of edges $\mathcal{M} = \bigcup_{F\in\mathcal{F}} M_F$, where $\mathcal{M}_F$ is a minimum matching in $G[Odd(G_{\mathcal{A}}) \cap V(F)]$ and $G_{\mathcal{A}} = \mathcal{A} \cup \mathcal{B}' \cup \mathcal{R}' \cup \mathcal{F}$ for each $F \in \mathcal{F}$.
6: Obtain a TSP tour T_3 in G using SHORTCUT.

section does not hold. Thus, only guessing $\mathcal{A}$ may not be enough to obtain a constant FPT approximation ratio. Since $|V_g|$ can also be arbitrarily larger than q, we cannot guess $\mathcal{B}$ directly. To address these issues, we use an idea in (Zhou, Zhang, and Guo 2025) to guess a set of limbs $\mathcal{B}'$ in FPT time by finding a minimum spanning $|\mathcal{A}|$-forest $\mathcal{F}$ in $G[V_g]$ and then guessing a set of anchors in a subset of $V(\mathcal{F})$, where $\mathcal{B}'$ is not necessarily equals to $\mathcal{B}$ but it satisfies $w(\mathcal{B}') \le w(\mathcal{B})$. Clearly,

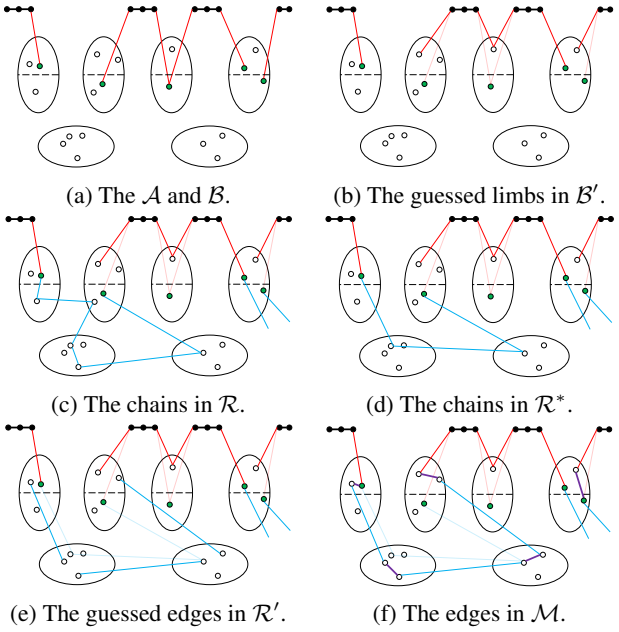
$$|\mathcal{F}| \le q \quad \text{and} \quad w(\mathcal{F}) \le w(\mathcal{R}). \tag{3}$$

Then, we augment $\mathcal{A}\cup\mathcal{B}'\cup\mathcal{F}$ into an Eulerian graph. The set of good chains $\mathcal{R}$ ensures the existence of a set of edges $\mathcal{R}'$ with $w(\mathcal{R}') \le w(\mathcal{R})$ such that $G_{\mathcal{A}} = \mathcal{A}\cup\mathcal{B}'\cup\mathcal{R}'\cup\mathcal{F}$ is a connected graph and $|Odd(G_{\mathcal{A}}) \cap V(F)|$ is even for each $F \in \mathcal{F}$. Thus, we can find a minimum matching $\mathcal{M}_F$ in $G[Odd(G_{\mathcal{A}})\cap V(F)]$ for each $F \in \mathcal{F}$ to fix the parity of the odd-degree vertices in $G_{\mathcal{A}}$. We will also prove $w(M_F) \le w(F)$. Let $\mathcal{M} = \bigcup_{F\in\mathcal{F}} M_F$. Finally, we obtain an Eulerian graph $G'_{\mathcal{A}} = \mathcal{A}\cup\mathcal{B}'\cup\mathcal{R}'\cup\mathcal{F}\cup\mathcal{M}$, and then we take shortcuts on $G'_{\mathcal{A}}$ to obtain a TSP tour without increasing the weight. Thus, by (1), the tour has weight at most

$$\begin{aligned} w(G'_{\mathcal{A}}) &\le w(\mathcal{A}) + w(\mathcal{B}') + w(\mathcal{R}') + w(\mathcal{F}) + w(\mathcal{M}) \\ &\le w(\mathcal{A}) + w(\mathcal{B}) + w(\mathcal{R}) + 2w(\mathcal{R}) \le 3 \cdot \text{OPT} \end{aligned}$$

ALG.3 is described in Algorithm 3, where LIMB, CONNECT, and SHORTCUT are sub-algorithms explained later. An illustration of our ALG.3 can be found in Fig. 3.

We remark that, instead of guessing a set of limbs $\mathcal{B}'$, the previous FPT 11-approximation algorithm (Zhou, Zhang, and Guo 2025) mainly uses the spanning $|\mathcal{A}|$-forest $\mathcal{F}$ to guess a set of edges E' such that $\mathcal{A} \cup E'$ forms a TSP tour in $G[V_b \cup V(E')]$. Then, it constructs a special spanning forest $\mathcal{F}'$ and obtains a TSP tour by taking shortcuts on $\mathcal{A}\cup E'\cup 2\mathcal{F}'$, similar to the FPT 3-approximation algorithm for TSP parameterized by p, as mentioned earlier.



(a) The $\mathcal{A}$ and $\mathcal{B}$. (b) The guessed limbs in $\mathcal{B}'$. (c) The chains in $\mathcal{R}$. (d) The chains in $\mathcal{R}^*$. (e) The guessed edges in $\mathcal{R}'$. (f) The edges in $\mathcal{M}$.

Figure 3: An illustration of our ALG.3. Black (resp., white) nodes denote bad (resp., internal) vertices; green nodes are anchors. Each ellipse denotes a tree $F \in \mathcal{F}$, with the upper part indicating its potential set V_F. In (a), the edges in $E(\mathcal{A})$ (resp., limbs in $\mathcal{B}$) are shown in black (resp., red); In (b), the guessed limbs in $\mathcal{B}'$ are shown in red, and the white and green nodes incident to red edges denote guessed anchors; In (c), the edges in $E(\mathcal{R})$ are shown in blue; In (d), the edges in $E(\mathcal{R}^*)$ are shown in blue; In (e), the guessed edges in $\mathcal{R}'$ are shown in blue; In (f), the edges in $\mathcal{M}$ are shown in purple.

Next, we show the sub-algorithms LIMB and CONNECT in Steps 3 and 4, the property of $\mathcal{M}$ in Step 5, and the sub-algorithm SHORTCUT in Step 6, respectively.

The sub-algorithm: LIMB Given $\mathcal{A}$ and $\mathcal{F}$, we assume that $T^* = a_1...b_1...a_{|\mathcal{A}|}...b_{|\mathcal{A}|}...$, where $a_i...b_i \in \mathcal{A}$ for each i. Denote the number of anchors between b_i and a_{i+1} in T^* by f_i, where $f_i \in \{1, 2\}$. For each tree $F \in \mathcal{F}$, initialize a *potential* (vertex) set $V_F := \emptyset$. LIMB works as follows.

First, we guess the values of $f_1, ..., f_{|\mathcal{A}|}$, which specify the number of anchors between each pair (b_i, a_{i+1}) in T^*. This allows us to determine the total number and relative positions of anchors in the ordering $\overline{a_1...b_1...a_{|\mathcal{A}|}...b_{|\mathcal{A}|}...}$. For each such anchor, we can determine whether it is a single or pair anchor and how many bad vertices it connects to in T^*. However, we do not know the exact identity of each anchor, i.e., which specific good vertex it corresponds to.

For example, consider the T^* in Fig. 1. If we correctly guess $f_1 = 2$, $f_2 = 1$, and $f_3 = 2$, then we know that the anchors, in order, are pair, pair, single, pair, and pair. The corresponding sets of bad vertices incident to them are $\{x_4\}$, $\{x_5\}$, $\{x_6, x_7\}$, $\{x_7\}$, $\{x_1\}$, respectively.

Next, for each anchor x, we guess the tree $F_x \in \mathcal{F}$ with $x \in V(F)$, select a set of potential vertices $V_x \subseteq V(F_x)$, and update the potential set by setting $V_{F_x} := V_{F_x} \cup V_x$. We

Algorithm 4: LIMB

Input: An instance $G = (V = V_g \cup V_b, E, w)$, $\mathcal{A}$, and $\mathcal{F}$.
Output: A set of limbs $\mathcal{B}'$.

Initialize $V_F \coloneqq \emptyset$ for each $F \in \mathcal{F}$, and $\mathcal{B}' = \emptyset$;
Guess the values of $f_1, ..., f_{|\mathcal{A}|}$.
for each anchor $x \in V_a$ in order $\overline{a_1...b_1...a_{|\mathcal{A}|}...b_{|\mathcal{A}|}...}$ **do**
    Guess the unique tree $F_x \in \mathcal{F}$ with $x \in V(F_x)$, and let $n_x = \min\{2q, |V(F_x)|\}$. ▷ We do not know the identity of x.
    if x is a single anchor **then** ▷ This can be determined by the f_i w.r.t. x.
        Let y and z be the bad vertices incident to x in T^*.
        Obtain a set of n_x vertices V_x by selecting $x' \in V(F_x)$ in the increasing order of $w(x', y) + w(x', z)$.
    else
        Let y be the unique bad vertex incident to x in T^*.
        Obtain a set of n_x vertices V_x by selecting $x' \in V(F_x)$ in the increasing order of $w(x', y)$.
    end if
    Update $V_{F_x} \coloneqq V_{F_x} \cup V_x$.
end for
for each anchor $x \in V_a$ in order $\overline{a_1...b_1...a_{|\mathcal{A}|}...b_{|\mathcal{A}|}...}$ **do**
    Guess x by enumerating a vertex $x' \in V_{F_x}$.
    if x is a single anchor **then**
        Let y and z be the bad vertices incident to x in T^*.
        Update $\mathcal{B}' \coloneqq \mathcal{B}' \cup \{x'y, x'z\}$. ▷ $x'y$ and $x'z$ are the limbs w.r.t. x'.
    else
        Let y be the unique bad vertices incident to x in T^*.
        Update $\mathcal{B}' \coloneqq \mathcal{B}' \cup \{x'y\}$. ▷ $x'y$ is the limb w.r.t. x'.
    end if
end for

will show that the size of each final potential set is $\mathcal{O}(q^2)$, and then, by guessing anchors in these potential vertex sets, we can obtain a set of limbs $\mathcal{B}'$ with desirable properties.

The details are put in Algorithm 1.

Lemma 4 (*). *There exists a configuration for the guessed anchors such that*

1. *the guessed anchors are pair-wise distinct;*
2. *for each anchor $x \in V_a$, the guessed anchor x' for x satisfies $x' = x$ if $x \in V_{F_x}$, and $x' \in V_x \subseteq V_{F_x}$ otherwise;*
3. *the set of all limbs $\mathcal{B}'$ w.r.t. the guessed anchors satisfies $w(\mathcal{B}') \leq w(\mathcal{B})$.*

Moreover, LIMB can compute such a configuration in $\mathcal{O}(2^q \cdot q^{2q} \cdot (4q^2)^{2q})$ guesses, where each guess takes $n^{\mathcal{O}(1)}$ time.

The sub-algorithm: CONNECT The graph $\mathcal{A} \cup \mathcal{B}' \cup \mathcal{F}$ may be disconnected. For example, any tree $F \in \mathcal{F}$ with $V(F) \cap V_a = \emptyset$ must form a component (see (b) in Fig.3).

We introduce the sub-algorithm CONNECT, which augments $\mathcal{A} \cup \mathcal{B}' \cup \mathcal{F}$ into a connected graph $\mathcal{A} \cup \mathcal{B}' \cup \mathcal{R}' \cup \mathcal{F}$ by guessing a set of edges $\mathcal{R}'$.

It is easy to verify that adding the full set of good chains $\mathcal{R}$ to $\mathcal{A} \cup \mathcal{B}' \cup \mathcal{F}$ yields a connected graph. However, since $|V(\mathcal{R})| = |V_g| = n - q$, enumerating all of $\mathcal{R}$ is not feasible in FPT time. Therefore, in CONNECT, we instead guess the subset of trees $\mathcal{F}_R \subseteq \mathcal{F}$ connected by each $R \in \mathcal{R}$, as $|\mathcal{F}| \leq q$ by (3). Note that this approach leads to $2^{\mathcal{O}(q^2)}$ guesses.

Algorithm 5: CONNECT

Input: An instance $G = (V = V_g \cup V_b, E, w)$, $\mathcal{A}$, $\mathcal{B}'$, and $\mathcal{F}$.
Output: A set of edges $\mathcal{R}'$.

Initialize $\mathcal{R}' = \emptyset$.
Obtain a complete graph $\widetilde{G} = (\widetilde{V}, \widetilde{E}, \widetilde{w})$ by contracting each $F \in \mathcal{F}$ into a vertex u_F, where $\widetilde{w}$ is defined as in (4).
Set $\widetilde{V}_a = \{u_F \mid F \in \mathcal{F}, V(F) \cap V_a \subseteq V_g^2, V(F) \setminus V_a \neq \emptyset\}$.
Guess the partition of $\widetilde{V}_a$, $\widetilde{\mathcal{V}} = \{\widetilde{V}_{R^*} \cap \widetilde{V}_a \mid R^* \in \mathcal{R}^*\}$, where $\widetilde{V}_{R^*} = \{u_F \mid F \in \mathcal{F}_{R^*}\}$, $\mathcal{F}_{R^*}$ is the set of trees in $\mathcal{F}$ connected by R^*, and R^* is obtained by shortcutting the internal vertices of $R \in \mathcal{R}$, ensuring that each vertex in $\widetilde{V}_a$ appears in only one of the paths in $\mathcal{R}^* = \{R^* \mid R \in \mathcal{R}, |V(R)| \geq 2\}$.
for each good chain $R \in \mathcal{R}$ with $V(R) \geq 2$ (in order T^*) **do**
    Denote the pair anchors w.r.t. R by x and x'. ▷ We do not know x and x' but we know F_x and $F_{x'}$ by $\mathcal{B}'$.
    Set $\widetilde{V}_{R^*} \coloneqq (\widetilde{V}_{R^*} \cap \widetilde{V}_a) \cup \{u_{F_x}, u_{F_{x'}}\}$.
    Using the DP (Bellman 1962), find a minimum TSP tour in $\widetilde{G}[\widetilde{V}_{R^*}]$ if $F_x = F_{x'}$, and a minimum TSP path between u_{F_x} and $u_{F_{x'}}$ in $\widetilde{G}[\widetilde{V}_{R^*}]$ otherwise.
    Let E_R be the corresponding set of edges in G w.r.t. the obtained tour or path.
    Update $\mathcal{R}' \coloneqq \mathcal{R}' \cup E_R$.
end for

To improve efficiency, we refine the strategy to reduce the number of guesses to $2^{\mathcal{O}(q \log q)}$.

First, we construct a complete graph $\widetilde{G} = (\widetilde{V}, \widetilde{E}, \widetilde{w})$ by contracting each tree $F \in \mathcal{F}$ into a vertex u_F, and define the weight between u_F and $u_{F'}$ as

$$\widetilde{w}(u_F, u_{F'}) = \min_{v \in V(F), v' \in V(F')} w(v, v'). \tag{4}$$

For each chain $R \in \mathcal{R}$, we obtain a minimal chain R_m connecting all trees in $\mathcal{F}_R$ by shortcutting its internal vertices. Then, we obtain a new chain R^* by shortcutting the internal vertices, ensuring that each vertex in $\widetilde{V}_a \coloneqq \{u_F \mid F \in \mathcal{F}, V(F) \cap V_a \subseteq V_g^2, V(F) \setminus V_a \neq \emptyset\}$ appears in only one path in $\mathcal{R}^* \coloneqq \{R^* \mid R \in \mathcal{R}, |V(R)| \geq 2\}$. It can be verified that the graph $\mathcal{A} \cup \mathcal{B}' \cup \mathcal{R}^* \cup \mathcal{F}$ remains connected.

Let $\mathcal{F}_{R^*}$ denote the set of trees in $\mathcal{F}$ connected by R^*, and define $\widetilde{V}_{R^*} \coloneqq \{u_F \mid F \in \mathcal{F}_{R^*}\}$. Then, the collection $\widetilde{\mathcal{V}} \coloneqq \{\widetilde{V}_{R^*} \cap \widetilde{V}_a \mid R^* \in \mathcal{R}^*\}$ forms a partition of $\widetilde{V}_a$, which can be guessed in $2^{\mathcal{O}(q \log q)}$ times, since $\widetilde{V}_a$ is determined from $\mathcal{B}'$ and satisfies $|\widetilde{V}_a| \leq |\mathcal{F}| \leq q$ by (3).

Once $\widetilde{\mathcal{V}}$ is guessed, we obtain $\widetilde{V}_{R^*} \cap \widetilde{V}_a$ for each $R \in \mathcal{R}$ with $|V(R)| \geq 2$. Moreover, since $\widetilde{V}_{R^*} \setminus \widetilde{V}_a$ corresponds to trees in $\mathcal{F}$ containing the anchors of R^*, it is also determined by $\mathcal{B}'$. So, we obtain $\widetilde{V}_{R^*}$ as $\widetilde{V}_{R^*} = (\widetilde{V}_{R^*} \cap \widetilde{V}_a) \cup (\widetilde{V}_{R^*} \setminus \widetilde{V}_a)$.

We then compute an optimal TSP path or tour in $\widetilde{G}[\widetilde{V}_{R^*}]$ using the DP method (Bellman 1962), which yields a set of edges in G, denoted by E_R, connecting all trees in $\mathcal{F}_{R^*}$ (see (e) in Fig.3). By construction, $w(E_R) \leq w(R^*) \leq w(R)$.

Finally, we define $\mathcal{R}' = \bigcup_{R \in \mathcal{R}} E_R$, where $E_R = \emptyset$ if $|V(R)| = 1$.

The details are described in Algorithm 5.

Lemma 5 (*). *CONNECT can compute $\mathcal{R}'$, a set of edges between good vertices, in $2^{\mathcal{O}(q \log q)}$ guesses such that*

1. $\mathcal{A} \cup \mathcal{B}' \cup \mathcal{R}' \cup \mathcal{F}$ *forms a connected graph* $G_{\mathcal{A}}$*;*
2. $|Odd(G_{\mathcal{A}}) \cap V(F)|$ *is even for each* $F \in \mathcal{F}$*;*
3. $Odd(G_{\mathcal{A}}) \cap V_b = \emptyset$*;*
4. $w(\mathcal{R}') \leq w(\mathcal{R})$.

Moreover, each guess takes $\mathcal{O}(q^2 \cdot 2^q) \cdot n^{\mathcal{O}(1)}$ *time.*

The sub-algorithm: SHORTCUT Given $G_{\mathcal{A}} = \mathcal{A} \cup \mathcal{B}' \cup \mathcal{R}' \cup \mathcal{F}$, Step 5 of ALG.3 constructs an Eulerian graph $G'_{\mathcal{A}} = \mathcal{A} \cup \mathcal{B}' \cup \mathcal{R}' \cup \mathcal{F} \cup \mathcal{M}$ by using $\mathcal{M}$ to fix the degree parity of the vertices in $Odd(G_{\mathcal{A}})$. By Lemma 5, this step is feasible.

To analyze the quality of $\mathcal{M}$, we use the following lemma.

Lemma 6 (*). *It holds that* $w(\mathcal{M}) \leq w(\mathcal{F})$.

Since $G'_{\mathcal{A}}$ is Eulerian, we use the sub-algorithm SHORTCUT to take shortcuts on $G'_{\mathcal{A}}$ to obtain a TSP tour T_3 in G with a non-increasing weight.

In $G'_{\mathcal{A}}$, each guessed single anchor connects exactly two bad chains. Since a triangle containing one bad vertex may violate the triangle inequality, in SHORTCUT, we first construct a new Eulerian graph $G''_{\mathcal{A}}$ by deleting some edges in $\mathcal{M} \cup \mathcal{F}$ and duplicating some guessed single anchors. Then, we can use a similar idea in Lemma 3 to take shortcuts on $G''_{\mathcal{A}}$ to obtain T_3 with a non-increasing weight.

The graph $G''_{\mathcal{A}}$ is constructed as follows. For each $F \in \mathcal{F}$ where $V(F)$ includes a guessed single anchor, we consider two cases.

Case 1: If $V(F)$ consists only of guessed single anchors, then it contains neither internal vertices nor pair anchors, implying that no edge in $\mathcal{R}'$ connects to F. In this case, we simply delete all edges in $E(F) \cup M_F$, which does not affect connectivity. Each guessed single anchor is then labeled as a *marked bad vertex* (which is still a good vertex).

Case 2: If $V(F)$ contains at least one internal vertex or pair anchor, then there exists an edge in $\mathcal{R}'$ or a limb in $\mathcal{B}'$ connecting F. For each guessed single anchor $x \in V(F)$, let its associated limbs be xy and xz, where y, z are bad vertices. We create a copy x' of x, remove xy and xz, and add $x'y$ and $x'z$. We refer to x' as a marked bad vertex.

These two cases are illustrated in Fig. 4 and the detailed procedure is given in Algorithm 6.

Lemma 7 (*). *Given* $G''_{\mathcal{A}}$*, there is an* $\mathcal{O}(n^2)$*-time algorithm to obtain a TSP tour* T_3 *in* G *with weight at most* $w(G''_{\mathcal{A}})$.

Theorem 3 (*). *For TSP parameterized by* q*, ALG.3 has an FPT 3-approximation ratio with runtime* $2^{\mathcal{O}(q \log q)} \cdot n^{\mathcal{O}(1)}$.

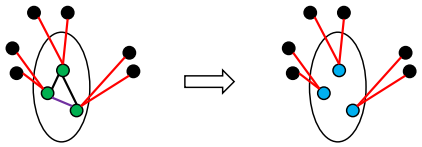
Conclusion

Many problems admit good approximation algorithms on metric graphs but become hard to approximate on general graphs. In practice, due to measurement errors in data or the intrinsic nature of certain problems, graphs that are originally metric may locally violate the triangle inequality. Although such violations are typically rare, they may render algorithms designed for metric graphs inapplicable. Recently, there has been growing interest in characterizing "distance" parameters between general and metric graphs, and in leveraging these parameters to develop parameterized algorithms.

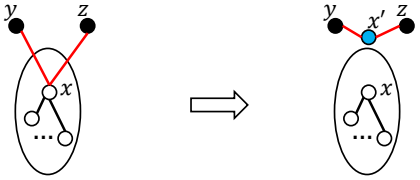
We investigate FPT approximation algorithms for the classic TSP problem under two parameters, p and q. By introducing new techniques, we significantly improve upon the best-known FPT approximation algorithms for both parameters. Our methods also show promise for extension to related problems parameterized by p and q, such as the asymmetric TSP (Traub and Vygen 2022) and the prize-collecting TSP (Blauth and Nägele 2023).

Several directions are worthy of further investigation:

1. When the parameter p (or q) is close to zero, our current approximation guarantees do not match the approximation ratio α achievable for metric TSP. Can this gap be closed? More generally, for the parameter q, is it possible to design a constant-factor approximation algorithm with runtime $2^{\mathcal{O}(q)} \cdot n^{\mathcal{O}(1)}$?
2. It would also be valuable to explore additional parameters that quantify the "distance" from a general graph to a metric one, potentially enabling better algorithms for handling near-metric graphs.



(a) $V(F)$ consists only of guessed single anchors.



(b) $V(F)$ contains one guessed single anchor and at least one internal vertex or pair anchor.

Figure 4: An illustration of the two cases. Each black node denotes a bad vertex, each green or white node denotes a good vertex, each blue node denotes a marked bad vertex; the edges in $E(F)$ are shown in black, the edges in $\mathcal{M}_F$ are shown in purple, and the limbs are shown in red. In (a), the black and purple edges are deleted; In (b), x' is a copy of x.

Algorithm 6: SHORTCUT

Input: $G = (V = V_g \cup V_b, E, w)$, $\mathcal{A}$, $\mathcal{B}'$, $\mathcal{R}'$, $\mathcal{F}$, and $\mathcal{M}$.
Output: A TSP tour T_3 in G.

for each $F \in \mathcal{F}$ such that $V(F)$ contains one guessed single anchor **do**
  if $V(F)$ consists only of guessed single anchors **then**
    Delete all edges in $E(F) \cup M_F$, and refer to all guessed single anchors as marked bad vertices.
  else
    for each guessed single anchor $x \in V(F)$ **do**
      Denote the limbs w.r.t. x by xy and xz.
      Create a marked bad vertex x'.
      Replace edges xy and xz with $x'y$ and $x'z$.
    end for
  end if
end for
Denote the above graph by $G''_{\mathcal{A}}$.
Obtain a TSP tour T_3 in G (see Lemma 7).

Acknowledgments

The work is supported by the National Natural Science Foundation of China under the grants 62502078 and 62372095, and by the Postdoctoral Fellowship Program of CPSF under Grant Number GZC20251102.

References

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

Andreae, T.; and Bandelt, H.-J. 1995. Performance guarantees for approximation algorithms depending on parametrized triangle inequalities. *SIAM Journal on Discrete Mathematics*, 8(1): 1–16.

Applegate, D. L. 2006. *The traveling salesman problem: a computational study*, volume 17. Princeton University Press.

Bampis, E.; Escoffier, B.; and Xefteris, M. 2024. Improved FPT Approximation for Non-metric TSP. *CoRR*, abs/2407.08392.

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

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

Blauth, J.; and Nägele, M. 2023. An Improved Approximation Guarantee for Prize-Collecting TSP. In Saha, B.; and Servedio, R. A., eds., *STOC 2023*, 1848–1861. ACM.

Böckenhauer, H.-J.; Hromkovič, J.; Klasing, R.; Seibert, S.; and Unger, W. 2002. Towards the notion of stability of approximation for hard optimization tasks and the traveling salesman problem. *Theoretical Computer Science*, 285(1): 3–24.

Christofides, N. 2022. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. *Operations Research Forum*, 3(1).

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2022. *Introduction to algorithms*. MIT Press.

Cygan, M.; Fomin, F. V.; Kowalik, Ł.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized algorithms*. Springer.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and intractability*, volume 174. Freeman San Francisco.

Held, M.; and Karp, R. M. 1962. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1): 196–210.

Karlin, A. R.; Klein, N.; and Gharan, S. O. 2021. A (slightly) improved approximation algorithm for metric TSP. In Khuller, S.; and Williams, V. V., eds., *STOC 2021*, 32–45. ACM.

Karlin, A. R.; Klein, N.; and Gharan, S. O. 2023. A Deterministic Better-than-3/2 Approximation Algorithm for Metric TSP. In Pia, A. D.; and Kaibel, V., eds., *IPCO 2023*, volume 13904, 261–274. Springer.

Khachay, M.; and Neznakhina, K. 2016. Approximability of the minimum-weight k-size cycle cover problem. *Journal of Global Optimization*, 66: 65–82.

Klasing, R.; and Mömke, T. 2018. A modern view on stability of approximation. In *Adventures Between Lower Bounds and Higher Altitudes: Essays Dedicated to Juraj Hromkovič on the Occasion of His 60th Birthday*, 393–408. Springer.

Lawler, E. 1976. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston.

Mohan, U.; Ramani, S.; and Mishra, S. 2017. Constant factor approximation algorithm for TSP satisfying a biased triangle inequality. *Theoretical Computer Science*, 657: 111–126.

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

Safilian, M.; Hashemi, S. M.; Eghbali, S.; and Safilian, A. 2016. An Approximation Algorithm for the Subpath Planning Problem. In *IJCAI 2016*, 669–675. IJCAI/AAAI Press.

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

Saller, S.; Koehler, J.; and Karrenbauer, A. 2025. A survey on approximability of traveling salesman problems using the TSP-T3CO definition scheme. *Annals of Operations Research*, 1–62.

Serdyukov, A. I. 1978. Some extremal bypasses in graphs. *Upravlyaemye Sistemy*, 17: 76–79.

Sumita, H.; Yonebayashi, Y.; Kakimura, N.; and Kawarabayashi, K. 2017. An Improved Approximation Algorithm for the Subpath Planning Problem and Its Generalization. In Sierra, C., ed., *IJCAI 2017*, 4412–4418. ijcai.org.

Traub, V.; and Vygen, J. 2022. An Improved Approximation Algorithm for The Asymmetric Traveling Salesman Problem. *SIAM Journal on Computing*, 51(1): 139–173.

Traub, V.; and Vygen, J. 2025. *Approximation algorithms for traveling salesman problems*. Cambridge University Press.

Zhao, J.; Sheng, Z.; and Xiao, M. 2025. Improved FPT Approximation Algorithms for TSP. *CoRR*, abs/2503.03642.

Zhao, J.; Xiao, M.; Peng, J.; and Xiong, Z. 2025. Improved Approximation Algorithms for Clustered TSP and Subgroup Planning. In *AAAI 2025*, 26742–26749. AAAI Press.

Zhou, J.; Li, P.; and Guo, J. 2022. Parameterized Approximation Algorithms for TSP. In *ISAAC 2022*, volume 248 of *LIPIcs*, 50:1–50:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Zhou, J.; Zhang, Z.; and Guo, J. 2025. An FPT Factor-11 Approximation Algorithm for TSP. In *COCOON 2025*, volume 15984 of *Lecture Notes in Computer Science*, 283–296. Springer.

Zhou, J.; Zhang, Z.; Wen, Y.; and Guo, J. 2025. From Metric to General Graphs: FPT Constant-Factor Approximation Algorithms for Three Location Problems. In *COCOON 2025*, volume 15984 of *Lecture Notes in Computer Science*, 297–310. Springer.