Approximating prize-collecting variants of TSP

Morteza Alimi, Tobias Mömke, Michael Ruderer

Angaben zur Veröffentlichung / Publication details:

Alimi, Morteza, Tobias Mömke, and Michael Ruderer. 2025. “Approximating prize-collecting variants of TSP.” In *50th International Symposium on Mathematical Foundations of Computer Science (MFCS 2025), Warsaw, Poland, August 25–29, 2025*, edited by Paweł Pawrychowski, Filip Mazowiecki, and Michał Skrzypczak, 7:1–17. Wadern: Schloss Dagstuhl –Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.MFCS.2025.7.

Approximating Prize-Collecting Variants of TSP

Morteza Alimi
University of Augsburg, Germany

Tobias Mömke
University of Augsburg, Germany

Michael Ruderer
University of Augsburg, Germany

Abstract

We present an approximation algorithm for the *Prize-collecting Ordered Traveling Salesman Problem* (PCOTSP), which simultaneously generalizes the Prize-collecting TSP and the Ordered TSP. The Prize-collecting TSP is well-studied and has a long history, with the current best approximation factor slightly below 1.6, shown by Blauth, Klein and Nägele [IPCO 2024]. The best approximation ratio for Ordered TSP is $\frac{3}{2} + \frac{1}{e}$, presented by Böhm, Friggstad, Mömke, Spoerhase [SODA 2025] and Armbruster, Mnich, Nägele [Approx 2024]. The former also present a factor 2.2131 approximation algorithm for Multi-Path-TSP.

We present a 2.097-approximation algorithm for PCOTSP, which is, to the best of our knowledge, the first result for this problem. Key ideas in our approach are to sample a set of trees and then to probabilistically pick up some vertices, and to use the pruning ideas of Blauth, Klein, Nägele [IPCO 2024] on the sampled vertices. While the sampling probability of vertices for our problem is lower than for PCTSP, intuitively leaving less spare penalty to spend, we leverage the cycle structure induced by the sampled trees together with a simple combinatorial algorithm to bring the approximation factor below 2.1.

Our techniques extend to Prize-collecting Multi-Path TSP, building on results from Böhm, Friggstad, Mömke, Spoerhase [SODA 2025], leading to a 2.41-approximation.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation Algorithms, TSP

Digital Object Identifier 10.4230/LIPIcs.MFCS.2025.7

Related Version *Previous Version*: `https://arxiv.org/abs/2411.14994`

Funding Partially supported by DFG Grant 439522729 (Heisenberg-Grant).

1 Introduction

The metric Traveling Salesman Problem (TSP), which asks for a shortest closed tour (i.e., a Hamiltonian cycle) in a metric space (V, c), $c : V \times V \to \mathbb{Q}_+$ on an n element vertex set V visiting each vertex exactly once[1] is one of the most well-studied problems in combinatorial optimization in its various incarnations. Christofides [15] and Serdjukov [23] gave a simple $\frac{3}{2}$ approximation algorithm for symmetric (undirected) TSP; an approximation factor slightly below $\frac{3}{2}$ was provided by Karlin, Klein and Oveis Gharan [20].

The path versions of TSP have also been subject to intense study. This line of study led to a surprising outcome: Traub, Vygen, and Zenklusen [25] show that for any $\epsilon > 0$, there is a reduction from path-TSP to TSP which only loses ϵ in the approximation factor. In their

[1] Note that the problem can alternatively be defined as having a weighted graph as input, and seeking to find a shortest closed walk (or shortest Eulerian multi-subgraph) that visits each vertex at least once.

book, Traub and Vygen [24] simplify the reduction and use Multi-Path-TSP (see below) as a building block. The book gives a comprehensive overview of the aforementioned results and more, including asymmetric (directed) variants of TSP.

A more general and more practical version of the problem can be defined by allowing the tour to omit some cities by paying some additional penalty. This *prize collecting* paradigm has been intensely studied for various combinatorial optimization problems; see, e.g., [1, 2] for some recent results regarding Prize-collecting Steiner Forest and Steiner Tree Problems. As regards the Prize-collecting Traveling Salesman Problem (PCTSP), Bienstock et al. [7] give a 2.5-approximation for this problem. Goemans and Williamson [18] give a factor 2 approximation. The first group to break the 2 barrier was Archer et al. [3], who achieved a factor of 1.979. Goemans [17] observed that by carefully combining the two aforementioned algorithms, one can achieve an approximation factor of 1.91. Blauth and Nägele [9] gave a factor 1.774-approximation. Blauth, Klein, and Nägele [8] achieved a factor of 1.599.

Another important generalization of TSP is attained by enforcing some ordering on a subset of vertices. Formally, in the metric Ordered Traveling Salesman Problem (OTSP), we are given a metric graph $G = (V, E), c\colon E \to \mathbb{R}_+$ together with k terminals $O = \{o_1, \ldots, o_k\}$, and the objective is to find a shortest tour that visits the terminals in order. OTSP is closely related to vehicle routing problems with pickup and delivery and the dial-a-ride problem; see [22, 6, 14, 16, 19]. Any α-approximation for TSP can be utilized to give an $(\alpha + 1)$-approximation algorithm for OTSP by finding an α-approximate tour on $V \setminus O$ and combining it with the cycle on O [10]. Furthermore, there is a combinatorial $(2.5 - 2/k)$-approximation algorithm [11]. A substantial improvement in the approximation factor to $\frac{3}{2} + \frac{1}{e}$ was achieved by [13] and [4].

In Multi-Path Traveling Salesman Problem (Multi-Path-TSP), in addition to a weighted graph, we are given a list of $2k$ terminals $\mathcal{T} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$. The objective is to find k paths $P_i,\ 1 \le i \le k$ of minimum total length, covering all vertices of the graph (see also Section 16.4 of [24]). Böhm, Friggstad, Mömke and Spoerhase [13] give a factor 2.2131 approximation algorithm for the problem.

1.1 Our Contribution and Overview of Techniques

As mentioned above, the prize-collecting setting is a standard setting for studying algorithmic problems. Theoretically, this is closely related to the Lagrangian relaxation paradigm, where constraints are allowed to be violated by paying some penalty. It is also natural from a practical perspective, where it is often possible to refuse to cover some entity if it incurs too high a cost, paying them some predefined penalty instead. In this article, we study OTSP and Multi-Path-TSP in this setting, which can also be viewed as generalizations of PCTSP.

▶ **Definition 1.** *In the Prize-collecting Ordered Traveling Salesman Problem (PCOTSP) we are given a metric weighted complete graph with penalties for vertices $G = (V, E),\ c\colon E \to \mathbb{R}_+, \pi : V \to \mathbb{R}_+$, together with k terminals $O = \{o_1, \ldots, o_k\}$. The objective is to find a tour C traversing $o_1, \ldots, o_k$ in order, that minimizes $c(C) + \sum_{v \notin V(C)} \pi(v)$.*

▶ **Definition 2.** *In the Prize-collecting Multi-Path Traveling Salesman Problem (PC-Multi-Path-TSP), we are given a metric weighted complete graph with vertex penalties $G = (V, E),\ c\colon E \to \mathbb{R}_+, \pi\colon V \to \mathbb{R}_+$, and a set of k terminal pairs $\mathcal{T} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$. The objective is to find k paths $P_i, 1 \le i \le k$, such that P_i is a path with endpoints s_i, t_i, minimizing $\sum_{i=1}^{k} c(P_i) + \sum_{v \notin (\cup V(P_i))} \pi(v)$.*[2]

[2] Similar to Path-TSP, we consider the k terminal pairs in Multi-Path TSP as part of what defines the

▶ **Theorem 3.** *There is a* 2.097*-approximation algorithm for PCOTSP.*

▶ **Theorem 4.** *There is a* 2.41*-approximation algorithm for PC-Multi-Path-TSP.*

In the remainder of this section we provide an overview of our algorithms and techniques for PCOTSP and PC-Multi-Path-TSP. The details of the algorithms and the proofs of correctness are covered in the following sections.

In Section 2, we specify the LP relaxation (OLP) of PCOTSP. Our algorithm first solves the LP, and then it samples a tree T_i for each of the k parts of the LP solution, in the manner of Lemma 2 in [8], and Theorem 4 in [13], which are both based on a result of Bang-Jensen, Frank and Jackson [5] and Post and Swamy [21][3] (see Lemma 6). The expected cost of each tree is at most the corresponding part of the LP solution, and its expected coverage is at least that of the corresponding y values. Now a key idea in our algorithm is to *even out* the penalty ratios paid by the algorithm. This is implemented in two ways.

1. Probabilistically *picking up* some of the vertices left out of the sampled trees (inevitably increasing the connection cost). This is achieved through a *pick-up threshold*, which determines the vertices which should be picked up (with a suitable probability), based on the LP solution.
2. Utilizing the spare penalty ratios of sampled vertices to bring down the cost of parity correction.

The second point is realized through an adaptation of the *pruning* idea from [8] to probabilistically prune away portions of the resultant structure with low connectivity. This allows us to assign weights to the edges of the tree, which, when combined with a suitable multiple of the LP solution, gives a point in the Q-join polytope, where Q is the set of odd degree vertices of the current structure. This allows for cheaper parity correction, because the tree edges with higher assigned weights, which are associated with lower value cuts, are pruned with higher probability, and thus the expected cost incurred by them remains low.

The prominent difference between our problem and the standard PCTSP, which necessitates finding new ideas for pruning, is the sampling probability for vertices. In PCTSP, the sampling probability for each vertex v is equal to y_v, hence the expected penalty ratio for v is one, and intuitively, there is lots of spare penalty for v to utilize for pruning. But in PCOTSP, the sampling probability of a vertex v is lower (bounded from below by $1 - e^{-y_v}$); hence there is little spare penalty ratio left (and for fewer vertices) to utilize for pruning. We obtain an improved adjustment of the pruning by combining our algorithm with a version of the classical algorithm for OTSP [10], which combines the cycle on terminals with a $\frac{3}{2}$-approximation for TSP on the remaining vertices. If the length of the cycle on terminals is low, this algorithm already gives a good approximation ratio. If the length of the cycle is large, this fact can be utilized to improve the analysis for parity correction, because we can show that there is no need to assign (nonzero) weights to the cycle edges.

We believe that this view of *evening out* the penalty ratio of vertices vis-a-vis the optimal LP solution is a useful conceptual tool for approaching prize collecting versions of TSP or other combinatorial optimization problems. In the case of PCOTSP, it is partly realized through the idea of probabilistically including in the solution those vertices which currently have a too high penalty ratio (based on the desired approximation factor). Squeezing out the

problem which cannot be relaxed. This is also consistent with standard practical applications where e.g. the start and finish depots cannot be dropped. Hence in PC-Multi-Path-TSP and its special case, PCOTSP, we assume the penalties for terminals are ∞. The techniques in this paper do not seem to extend directly to the case where terminals are allowed to be dropped.

[3] Post and Swamy showed that the Bang-Jensen, Frank, Jackson decomposition can be computed in polynomial time.

slack penalty ratio (again, based on the target approximation factor), is achieved through pruning. Here, the specific structure of the OTSP, and the partially constructed solution proves useful; we can show that if the current cycle through the terminals is small, combining the cycle on terminals with the best PCTSP approximation for the rest of vertices gives a good factor, while a large cycle improves the cost of pruning.

In the case of Multi-Path-TSP, Böhm, Friggstad, Mömke, and Spoerhase [13] propose two algorithms, and show that a careful combination of them leads to a good approximation. The first one, which doubles all edges except those on s_i, t_i paths (and is good when the sum of s_i, t_i distances is large) can be adapted to our setting by probabilistically picking up vertices with high penalty ratio, as opposed to picking up all remaining vertices in [13]. We replace their second algorithm, which finds a minimum length forest in which each terminal appears in exactly one of the components and then adds direct s_i, t_i edges with an algorithm that uses Lemma 6 to sample a tree from the related PCTSP, and then adds direct s_i, t_i edges.

2 Prize-collecting Ordered TSP

In this section we describe our algorithm for PCOTSP. Some of the technical proofs will be presented in the following sections.

2.1 Preliminaries and Definitions

In the OTSP, a tour can be decomposed into k paths, between o_i and o_{i+1}.[4] We can take the polyhedron determined by the following inequalities as the relaxation of one such path between two vertices s and t.[5] For each vertex v, the variable y_v indicates its fractional degree in the stroll.

$$\begin{aligned} y_s = y_t &= \tfrac{1}{2} \\ x(\delta(v)) &= 2y_v && \forall v \in V \\ x(\delta(S)) &\geq 1 && \forall S \subseteq V \setminus \{t\}, s \in S \qquad \textbf{(s-t-stroll relaxation)} \\ x(\delta(S)) &\geq 2y_v && \forall v \in S \subseteq V \setminus \{s,t\} \\ x, y &\geq 0 \end{aligned}$$

Now, the PCOTSP can be modeled as the following linear program. For $i = 1, \dots k$ we define x_i and y_i to be the vectors $(x_{i,e})_{e \in E}$ and $(y_{i,v})_{v \in V}$, respectively. For each i, the vector (x_i, y_i) is constrained to be a feasible point in the o_i-o_{i+1}-stroll relaxation, the sum $y_v = \sum_{i=1}^{k} y_{i,v}$ over all fractional degrees of a vertex v is an indicator of to what degree v is used in the solution. When v is not fully used, i.e., when $y_v < 1$, the LP has to pay a fractional penalty of $\pi_v(1 - y_v)$. We will usually refer to the pair (o_i, o_{i+1}) as (s_i, t_i), to emphasize that (x_i, y_i) is a fractional tour/stroll from o_i to o_{i+1}.

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} \sum_{i=1}^{k} c_e x_{i,e} + \sum_{v \in V} \pi_v (1 - y_v) \\ \text{subject to} \quad & y_v = \sum_{i=1}^{k} y_{i,v} && \forall v \in V \\ & (x_i, y_i) \text{ lies in the } s_i\text{-}t_i\text{-stroll relaxation} && \forall i \in [k] \end{aligned} \tag{OLP}$$

[4] For notational convenience, we identify o_{k+1} with o_1.
[5] Similar to the LP of [4]; see also [12, 8].

Note that every feasible solution to (OLP) has $y_o = 1$ for all terminals o. Similar to y_v, we will use x_e as a shorthand for $\sum_{i=1}^{k} x_{i,e}$. It follows immediately from the LP constraints that $x(\delta(S)) \geq 2y_v$ for any $v \in S \subseteq V \setminus O$. By contracting all terminals $o \in O$ into a single vertex r, we therefore obtain the relaxation of the normal PCTSP from [8], which also involves a root, which (without loss of generality; see, e.g., [3]) is required to be in the tour. Given a solution (x, y) to (OLP) and some threshold $\rho \in [0, 1]$, we define $V_\rho \coloneqq \{v \in V \mid y_v \geq \rho\}$.

2.2 A simple algorithm

We consider the following simple algorithm for PCOTSP, inspired by [10]. First, directly connect the terminals $o_1, \ldots, o_k$ in order, creating a simple cycle $\hat{C}$. Then, compute a solution to the PCTSP on the same instance. Since all terminals have infinite penalty, every tour T obtained in this way includes all terminals. We obtain an ordered tour of cost $c(T) + c(\hat{C})$ by following the original cycle $\hat{C}$ and grafting in the tour at an arbitrary terminal $o \in O$.

Using an $\hat{\alpha}$-approximation algorithm to solve the PCTSP, we know that the sum of tour- and penalty costs for this solution is at most $\hat{\alpha} \cdot \mathrm{opt}_{\mathrm{PCTSP}} \leq \hat{\alpha} \cdot \mathrm{opt}_{\mathrm{PCOTSP}}$. Since $c(\hat{C}) \leq \mathrm{opt}_{\mathrm{PCOTSP}}$, this immediately implies a $(1+\hat{\alpha})$-approximation algorithm for PCOTSP. One can see that this algorithm performs even better if we can guarantee that $\hat{C}$ is small. To be precise, for any $\alpha \geq \hat{\alpha}$ we obtain an α-approximation provided that $c(\hat{C}) \leq (\alpha - \hat{\alpha})\mathrm{opt}_{\mathrm{PCOTSP}}$. We therefore may assume $c(\hat{C}) \geq (\alpha - \hat{\alpha})\mathrm{opt}_{\mathrm{PCOTSP}}$ in the analysis of our main algorithm. The currently best value for $\hat{\alpha}$ is the approximation guarantee of (roughly) 1.599 obtained by [8].

2.3 Overview of our main algorithm

Fix $\alpha = 2.097$. We first solve (OLP)[6], to get an optimal solution (x^*, y^*). Using the following Lemma 5, we then *split off* the vertices v for which $y_v \leq \theta$, for a parameter θ to be determined later, to get a solution $(\hat{x}, \hat{y})$ for the LP were the remaining vertices have a certain minimum connectivity to the terminals. Vertices that have been split off will not be used in our final tour. Instead, we pay the full penalty for them. We state the following lemma for PCOTSP, as the proof is identical to the one for PCTSP.

▶ **Lemma 5** (Splitting off [8]). *Let (x, y) be a feasible solution to the PCOTSP relaxation and let $S \subseteq V \setminus O$. Then we can efficiently compute another feasible solution (x', y') in which $y'_v = 0$ for all $v \in S$, but $y'_v = y_v$ for all $v \notin S$, and $c(x') \leq c(x)$.*

Lemma 5 ensures that we can remove vertices from the support of x, without increasing the cost of x. Since we only split off vertices for which y^*_v is relatively low, we can afford to pay the additional penalties if we set $\theta = 1 - \frac{1}{\alpha}$ (we will prove this in Section 3.1). We proceed by sampling a set of trees based on $(\hat{x}, \hat{y})$.

▶ **Lemma 6** ([8], following [5]). *Suppose (x, y) is a feasible point in the s-t-stroll relaxation. In polynomial time we can find a set of trees $\mathcal{T}$ and weights $(\mu(T))_{T\in\mathcal{T}}$ such that (i) $\sum_{T\in\mathcal{T}} \mu(T) = 1$; and (ii) $\sum_{T\in\mathcal{T}: e\in E(T)} \mu(T) \leq x_e \quad \forall e \in E$; and (iii) $\sum_{T\in\mathcal{T}: v\in V(T)} \mu(T) \geq y_v, \quad \forall v \in V \setminus \{s, t\}$; and (iv) all trees span s and t.*

Lemma 6 can be used to sample a random tree of expected cost at most $c(x)$ which contains each vertex v with probability at least y_v and is guaranteed to connect s to t. It is straightforward to see that this can be achieved by choosing each tree $T \in \mathcal{T}$ with probability $\mu(T)$.

[6] The separation oracle for the LP boils down to separating subtour elimination constraints. Hence the LP can be solved in polynomial time using the ellipsoid algorithm.

For each component $(\hat{x}_i, \hat{y}_i)$ of our solution, we apply Lemma 6 and sample a tree T_i as we have described. Define $T := \dot{\bigcup}_{i=1}^{k} T_i$. Note that T is no longer a tree, and it might even have repeated edges.

2.3.1 Pruning and Picking up

To get some intuition on the the following steps, suppose that we had to pay the penalty for a vertex v if and only if v has not been sampled in T. In Lemma 10, we will show that the probability for this is at most $e^{-y_v^*}$, which gives us a bound of $e^{-y_v^*}\pi_v$ on the expected penalty cost of v. The LP however pays a $(1-y_v^*)$-fraction of π_v. We thus define the *penalty ratio* ρ_v of a vertex v as $\rho_v = \frac{e^{-y_v^*}}{1-y_v^*}$ and define σ_0 as unique solution to the equation $\alpha(1-\sigma_0) = e^{-\sigma_0}$. With this, if $y_v^* = \sigma_0$, we have $\rho_v = \alpha$, i.e., our expected penalty for v is at most α times the LP penalty.

Note that in the PCTSP regime, e.g., in the result of [8], the probability that a vertex v is in the (one) sampled tree is exactly y_v^*, which means that $\rho_v = 1 < \alpha$ for every v. But in our setting, the probability of a vertex not being sampled is [bounded from above by] $e^{-y_v^*}$, which intuitively means that the spare penalties are much more constrained, and the gains would be more meager. Nonetheless, we show how to utilize the specific structure of our problem to gain almost as much from pruning as in the setting of PCTSP.

We will now describe how we deal with vertices whose y^*-value is smaller or larger than σ_0. First, we consider those vertices for which $y_v^* < \sigma_0$ (and thus $\rho_v < \alpha$). These vertices v have some *spare* penalty ratio, which we utilize to prune v with certain probability.

For our pruning step, we need the following definition from [8]:

▶ **Definition 7.** *For a fixed LP solution (x, y), a tree T, and a threshold $\gamma \in [0, 1]$, we define core(T, γ) as the inclusion-wise minimal subtree of T that spans all vertices in $V(T) \cap V_\gamma$.*

Algorithmically, core(T, γ) can be obtained by iteratively removing leaves v with $y_v < \gamma$. To prune a tree T_i simply means to replace it by $T_i' = \text{core}(T_i, \gamma)$. We emphasize that for pruning T_i, we do not consider the local penalty values $y_{i,v}$, but the *global* values y_v. Furthermore, our construction ensures that s_i as well as t_i are part of the pruned tree T_i' (remember that $y_{s_i}^* = y_{t_i}^* = 1$). We will draw the global pruning threshold γ according to a suitable distribution D_γ which is defined by the following cumulative probability function: $F_\gamma(y) = \Pr[\gamma \leq y] := \frac{1-\alpha(1-y)}{1-e^{-y}}$. We will prove in Section 3.1 that this ensures that the slack in the penalty ratios is fully utilized.

We continue with those vertices for which $y_v > \sigma_0$ and describe our pickup step. For vertices v with $y_v > \sigma_0$, the penalty ratio is greater than α. In fact, their penalty ratio becomes arbitrarily high as y_v^* approaches 1. Intuitively, this means that the probability of these vertices to not be sampled is too high. We call these vertices *critical* and define $V_{\sigma_0} = \{v \in V \mid y_v^* \geq \sigma_0\}$. We *pick up* these critical vertices with certain probability. That means, we will probabilistically select some unsampled critical vertices and connect them to our solution. We first describe how these vertices are selected and then give the details on how we connect them.

Note that by our observation on ρ_v, the chance for picking up a fixed unsampled critical vertex v should increase with the value of y_v^*. Our strategy is therefore the following: As we did for our pruning threshold, we will draw a global pickup threshold $\sigma \in [\sigma_0, 1]$ from a distribution D_σ. We then pick up all critical vertices whose y-value is above σ, i.e., all vertices in $V_\sigma \setminus V(T)$.

Algorithm 1 The Prize-collecting Ordered TSP Algorithm.

Input: An PCOTSP instance
1: Compute an optimal solution (x^*, y^*) of OLP
2: Split off vertices $v \in V$ with $y^*_v \leq \theta$
3: Sample k trees $T_1, \dots T_k$ independently using Lemma 6
4: Sample a pruning threshold $\gamma \sim D_\gamma$
5: Prune each T_i based on γ to get T'_i. Let $T' = \bigcup_{i=1}^{k} T'_i$
6: Sample a pickup threshold $\sigma \sim D_\sigma$
7: Pick up all unsampled vertices with $y_v \geq \sigma$ to obtain T''
8: Add a shortest odd(T'')-join J to T'' to get H
9: **return** an ordered tour obtained by shortcutting H

The distribution D_σ is determined by the cumulative probability function F_σ which we state in the following. Choosing σ according to this function will ensure that for any $v \in V_{\sigma_0}$ the probability of being picked up is just high enough so that the expected penalty paid for v is at most α times the fractional penalty $\pi_v(1 - y^*)$ paid by the LP solution. We will formally prove this fact in Section 3.1. For now, we simply define $F_\sigma(y) \coloneqq 1 - \frac{\alpha(1-y)}{e^{-y}}$. We remark that by definition of σ_0, we have $\frac{\alpha(1-y)}{e^{-y}} = \frac{e^y(1-y)}{e^{\sigma_0}(1-\sigma_0)}$, from which it is easy to verify that F_σ is indeed a cumulative probability function, i.e., $F_\sigma(\sigma_0) = 0$, $F_\sigma(1) = 1$ and F_σ is monotonously increasing on $[\sigma_0, 1]$. To properly describe how we connect $V_\sigma \setminus T$ to T, we use the following definition from [13].

▶ **Definition 8.** *Let $X \subseteq Y \subseteq V$. An X-rooted spanning forest of Y is a spanning forest of Y such that each of its connected components contains a vertex of X.*

A cheapest X-rooted spanning forest of Y can be efficiently computed by contracting X, computing a minimal spanning tree of Y in the contracted graph and then reversing the contraction.

For our pickup step, we buy the cheapest forest F_P that spans V_σ and is rooted in $V_\sigma \cap V(T)$. This forest F_P has two crucial properties: (i) after buying F_P, each vertex $v \in V_\sigma \setminus T$ will be connected to T; and (ii) F_P only spans vertices with $y^*_v \geq \sigma$. Property (ii) follows immediately from the definition of F_P, and property (i) follows from the fact that $O \subseteq V_\sigma \cap T$ for any choice of σ.

Finally, we remark that our pickup and pruning steps target the disjoint vertex sets $\{v \in V \mid y^*_v \geq \sigma_0\}$ and $\{v \in V \mid \theta \leq y^*_v < \sigma_0\}$, so neither step interferes with the other.

2.3.2 Obtaining an ordered tour

To turn T'' into a feasible tour, we first need to correct parities, i.e., we ensure that every vertex has an even degree. Let H be the graph that is obtained by adding a cheapest odd(T'')-join to T'', where odd(T'') is defined as the set of odd degree vertices of T''. Observe that initially, each tree T_i contains a path P_i between o_i and o_{i+1}. Since all terminals $o \in O$ have $y^*_o = 1$, all of these paths survive the pruning step. So the multigraph H still contains k edge-disjoint o_i–o_{i+1}–paths which, taken together, form a closed walk C. By removing C from H, we obtain a graph whose connected components have even degree and can thus be shortcut into cycles. Furthermore, since H was connected, each of these cycles has a common vertex with C. Hence we can obtain a feasible tour of no greater cost than H by following the walk C and grafting in the cycles obtained by shortcutting the components of $H \setminus C$ on the way.

3 Analysis of Algorithm 1

In this section, we prove Theorem 3. We will compare the expected cost of our computed solution to the cost of the optimal LP solution (x^*, y^*). In particular, we compare the expected cost of our computed tour to the cost of x^* (Section 3.3) and the expected sum of our incurred penalties to the fractional penalty cost defined by y^* (Section 3.1). For a derandomized version of our algorithm, we refer to the appendix.

In the following, we use (x^*, y^*) to refer to the optimal LP solution computed by the algorithm, and (x, y) to refer to the LP solution after the splitting-off step.

3.1 Bounding the Penalty Ratios

In this subsection, we prove the following lemma:

▶ **Lemma 9.** *The expected total penalty cost paid by Algorithm 1 is at most*

$$\alpha \cdot \sum_{v \in V} \pi_v \cdot (1 - y^*_v).$$

We note that we can express the expected total penalty cost paid by our algorithm as $\sum_{v \in V} \pi_v \cdot \Pr\big[v \notin V(T'')\big]$. We can thus prove Lemma 9 by showing that for each vertex v, the ratio $\Pr\big[v \notin V(T'')\big]/(1 - y^*_v)$ is at most α.

Due to Step 2 in Algorithm 1, i.e., due to the splitting-off step, T'' spans only vertices whose y-value is at least θ. Vertices v with $y^*_v < \theta$ are therefore included in $V(T'')$ with probability 0. However, our choice of $\theta = 1 - \frac{1}{\alpha}$ guarantees that for the vertices v with $y^*_v < \theta$, we have $\Pr\big[v \notin V(T'')\big]/(1 - y^*_v) \leq 1/(1 - \theta) = \alpha$. To continue our analysis for the remaining vertices, i.e., those with $y^*_v \geq \theta$, we show the following lemma:

▶ **Lemma 10.** *Let v be a vertex with $y^*_v \geq \theta$. Then the probability that v is* not *in T is at most e^{-y^*}.*

Proof. By Lemma 6, the probability that v is not in T_i is at most $1 - y^*_{i,v}$ for any fixed $i \in [k]$. The probability that this happens for *all* $i \in [k]$ is at most

$$\prod_{i=1}^{k} \Pr[v \notin V(T_i)] \leq \prod_{i=1}^{k} (1 - y^*_{i,v}) \leq \exp\left(-\sum_{i=1}^{k} y^*_{i,v}\right) = e^{-y^*_v}. \qquad \blacktriangleleft$$

Note that the distributions from which σ and γ are chosen guarantee that $\theta \leq \gamma < \sigma_0 \leq \sigma \leq 1$, i.e., only vertices v with $y^*_v \in [\theta, \sigma_0)$ can be pruned and only vertices with $y^*_v \in [\sigma_0, 1]$ can be picked up.

Now consider a vertex v with $y^* \in [\theta, \sigma_0)$. There are two cases in which we have to pay v's penalty: if v is not sampled, and if v is sampled but immediately pruned afterwards.

By Lemma 10, the probability of the former is at most $e^{-y^*_v}$, whereas the probability of being pruned – given that v was sampled in the first place – can be bounded from above by $\Pr[\gamma > y^*_v] = 1 - F_\gamma(y^*_v)$.[7] Our choice of $F_\gamma(y) = \frac{1-\alpha(1-y)}{1-e^{-y}}$ guarantees that the expected

[7] The probability can be lower if one of the sampled trees has a vertex with higher y^*-value in the subtree rooted at v.

penalty is just high enough:

$$\frac{\Pr\left[v \notin V(T'')\right]}{1-y_v^*} \leq \frac{(1-\Pr[v \in T]) + \Pr[v \in T] \cdot (1-F_\gamma(y_v^*))}{1-y_v^*}$$
$$= \frac{1-\Pr[v \in T]F_\gamma(y_v^*)}{1-y_v^*} \leq \frac{1-(1-e^{-y_y^*})F_\gamma(y_v^*)}{1-y_v^*} = \alpha.$$

Finally, consider a vertex v with $y_v^* \in [\sigma_0, 1]$. This time, there is only one case in which we have to pay the penalty for v: when v is neither sampled, nor picked up afterwards. Again, the probability of not being sampled is at most $e^{-y_v^*}$ whereas the probability of *not* being picked up – given that v was not sampled previously – is $\Pr[\sigma > y_v^*] = 1 - F_\sigma(y_v^*)$. Again, our choice of $F_\sigma(y) = 1 - \frac{\alpha(1-y)}{e^{-y}}$ guarantees that

$$\frac{\Pr\left[v \notin V(T'')\right]}{1-y_v^*} \leq \frac{e^{-y_v^*} \cdot (1-F_\sigma(y_v^*))}{1-y_v^*} = \alpha.$$

This concludes our proof of Lemma 9.

3.2 Parity Correction

In order to analyze the expected cost of both the parity correction step and of our final tour, we first need to further investigate the structure of our computed solution and introduce some additional notation. Recall that T denotes the union of all sampled trees and that it includes the closed walk C, which is obtained by joining all s_i-t_i-paths $P_i \subseteq T_i$. Let R be the graph that contains all remaining edges, i.e., the (multi-)graph induced by $E(T) \setminus E(C)$.

While it is convenient to think of C as a cycle, note that the paths $P_1, \ldots, P_k$ are not necessarily vertex disjoint. However, we can guarantee that C is a Eulerian multigraph spanning all terminals. In the same sense, R can be thought of as a collection of small trees which are all rooted at the cycle C (in reality, some of those trees may intersect each other). In the following, we will call the edges of C *cycle edges* and the edges of R *tree edges*. A depiction of C and R can be seen in Fig. 1.

When we prune the trees T_i into T_i', the paths P_i are not affected (because the pruning step guarantees that o_i and o_{i+1} stay connected in T_i'). So our pruning step can only remove edges from R. We thus define $\text{core}(R, \gamma)$ as the graph that is obtained by pruning all trees in T with pruning threshold γ and then removing the cycle C. We now partition the edge set of R into layers. Intuitively, the i-th layer of R contains those edges that are contained in $\text{core}(R, \gamma)$ if and only if γ does not exceed some value η_i.

Let $\eta_1 > \ldots > \eta_\ell$ be the y^*-values of all vertices that might be affected by our pruning step, i.e., $\{\eta_1, \ldots, \eta_\ell\} = \{y_v \mid v \in V \text{ and } \theta \leq y_v \leq \sigma_0\} \cup \{\sigma_0\}$. By definition, we always have $\eta_1 = \sigma_0$. Now let $E_1 = \text{core}(T, \eta_1)$ and $E_i = \text{core}(T, \eta_i) \setminus E_{i-1}$ for $i = 2, \ldots, \ell$. One can see that $E_1 \cup E_2 \cup \ldots \cup E_\ell$ is a partitioning of $E(R)$.

To be able to bound the cost of the cheapest odd(T'')-join J , we define the following vector

$$z := \beta x + \underbrace{\sum_{i:\eta_i \geq \gamma} (1 - 2\eta_i\beta)\chi^{E_i}}_{z_\gamma} + \underbrace{\max\{0, 1-2\beta\sigma\}\chi^{F_P}}_{z_\sigma} \tag{1}$$

where $\beta = \frac{1}{3\sigma_0 - \theta}$. To simplify future arguments, we also define the two components z_γ and z_σ of z, as specified in (1). We remark that the vector $\beta x + z_\gamma$ has been used (for a different value of β) in [8]. In Section 3.3, the cost of z will be used as an upper bound for $c(J)$. To this end, we now prove the following lemma.

▶ **Lemma 11.** *z lies in the dominant of the odd(T'')-join polytope.*

Proof. First, observe that all coefficients z_e are non-negative. We now consider an arbitrary $S \subseteq V$ for which $|S \cap \text{odd}(H)|$ is odd and show that $z(\delta(S)) \geq 1$.

We begin with the case where S cuts the terminal set, i.e., where $0 < |S \cap \{o_1, \dots, o_k\}| < k$. In this case, S separates at least two terminal pairs (o_{i_1}, o_{i_1+1}) and (o_{i_2}, o_{i_2+1}), which implies $x(\delta(S)) \geq 2$ and therefore $z(\delta(S)) \geq \beta x(\delta(S)) \geq 1$. In the following we can thus assume that $\emptyset \neq S \subseteq V \setminus \{o_1, \dots, o_k\}$.

Now suppose that $\delta(S)$ contains a pickup edge $e = \{u, v\} \in E(F_P)$ s.t. $u \in S$ and $v \notin S$. Recall that F_P only spans vertices w with $y_w \geq \sigma \geq \sigma_0$. We therefore have $u \in S \subseteq V \setminus \{o_1, \dots, o_k\}$ which (by the connectivity constraints of (OLP)) implies $x(\delta(S)) \geq 2y_u \geq 2\sigma_0$ and therefore $z(\delta(S)) \geq \beta x(\delta(S)) + \max\{0, 1 - 2\beta\sigma_0\} \geq 1$.

It remains to show the bound for the case when $\delta(S)$ only contains cycle and tree edges. By a simple counting argument, one can see that $|\delta_{T''}(S)|$ has the same parity as $|S \cap \text{odd}(T'')|$ and therefore must be odd. Now observe that by our assumption, $|\delta_{F_P}(S)| = 0$ and that because C is a Eulerian multigraph, $|\delta_C(S)|$ must be even. If follows that $\delta(S)$ must have an odd number of tree edges. We finish our proof by marginally adapting an argument from [8].

First, we consider the case where $\delta(S)$ contains *exactly* one tree edge e. This is only possible if S includes a whole subtree T_e of one of the trees in T'. Because this subtree T_e has survived the pruning step, e must lie in some layer E_i for which $\eta_i \geq \gamma$. Furthermore, if $e \in E_i$, then T_e must contain at least one vertex v with $y_v \geq \eta_i$, which implies that $x(\delta(S)) \geq 2\eta_i$. So we have

$$z(\delta(S)) \geq \beta x(\delta(S)) + (1 - 2\eta_i) \geq 2\beta\eta_i + (1 - 2\beta\eta_i) \geq 1.$$

If, however, $\delta(S)$ contains at least three tree edges, we know that all vertices $v \in S$ have $y_v \geq \theta$ and that each tree edge contributes at least $(1 - 2\beta\sigma_0)$ to z_γ. Therefore $z(\delta(S)) \geq 2\beta\theta + 3(1 - 2\beta\sigma_0) \geq 1$. ◀

3.3 Bounding the Tour Cost

In this section, we show the following bound on the cost of our computed tour:

▶ **Lemma 12.** $\mathbb{E}\big[c(T'')\big] \leq \alpha \cdot c(x^*)$

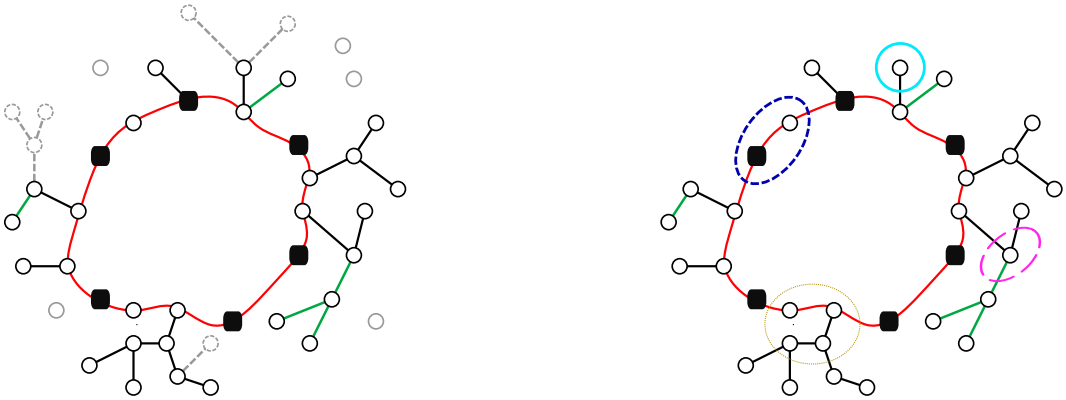
The total cost of our computed tour can be bounded from above by

$$\mathbb{E}\big[c(H)\big] \leq \mathbb{E}\big[c(T')\big] + \mathbb{E}\big[c(F_P)\big] + \mathbb{E}\big[c(J)\big].$$

Our bound on the cost of J is given by the cost of the parity correction vector $z = \beta x + z_\gamma + z_\sigma$. We start by analyzing the expected combined value of $c(T')$ and $c(z_\gamma)$.

▶ **Lemma 13.** $\mathbb{E}\big[c(T') + c(z_\gamma)\big] \leq c(x)\Big(2 + \hat{\alpha} - \alpha - (2 + 2\hat{\alpha})\beta\sigma_0 + 2\alpha\beta\sigma_0\Big)$ *where $\hat{\alpha}$ denotes the approximation guarantee of the current best PCTSP algorithm.*

The main idea of the proof of Lemma 13 is that the weight which each layer E_i is assigned in z_γ is large when η_i is small and vice versa. At the same time, the chance for layer E_i to be present after pruning and thus to contribute at all to both z_γ and $c(T')$ is an increasing function of η_i. By balancing out these two values, we obtain an upper bound or the contribution of all layers in R to $c(T') + c(z_\gamma)$. At the same time we take into account that the cycle C only contributes toward $c(T')$ but crucially not towards the cost of the



(a) (b)

Figure 1 (a) The graph T'' after pruning and picking up critical vertices. The terminals in O are drawn as black rectangles. The cycle C is depicted in red, the surviving edges of R are drawn in black and edges in F_P in green. The greyed out vertices and edges do not belong to T''. They have either been pruned (the dashed vertices and edges), not sampled, or split off. (b) The same graph T'' with the various cuts that are considered in the proof of Lemma 11 drawn in different colors. The dashed blue cut is an example for the case where $0 < |S \cap O| < k$. The dashed pink cut shows the case where a pickup edge (drawn in green) is cut. Note that even though the vertex in S was not picked up itself, it is still part of F_P and thus must have a high y-value. The remaining two cuts (drawn in brown and light blue) show the two cases where $\delta(S)$ contains an odd number of tree edges.

parity correction vector. This is where we utilize our assumption that $\hat{C}$ and therefore also C can be lower bounded by a constant fraction of opt. For formally proving Lemma 13, we need the following technical Lemma 14.

▶ **Lemma 14.** *For $\frac{2}{3} \leq \sigma_0 \leq 1$, the function $g(y) = Pr[\gamma \leq y](2 - 2\beta y)$ attains its maximum value over the interval $[\theta, \sigma_0]$ at $y = \sigma_0$.*

Proof. First, observe that $\Pr[\gamma \leq y] = F_\gamma(y) = \frac{1-\alpha(1-y)}{1-e^{-y}}$ and therefore $g(y) = 2F_\gamma(y)(1-\beta y)$. The claim follows from proving that g is monotonously increasing on $[\theta, \sigma_0]$ by showing that its derivative is strictly positive:

$$\begin{aligned}\frac{1}{2}\frac{d}{dy}g(y) &= f_\gamma(y)(1-\beta y) - \beta F_\gamma(y) \geq f_\gamma(y)(1-\beta\sigma_0) - \beta \\ &\geq \alpha(1-\beta\sigma_0) - \beta \qquad (2)\\ &> 0, \qquad (3)\end{aligned}$$

where f_γ denotes the density function of γ. In (2) we used that for $y \in [\theta, \sigma_0]$

$$\begin{aligned}f_\gamma(y) &= \frac{\alpha}{1-e^{-y}} - (1-\alpha(1-y))\frac{e^{-y}}{(1-e^{-y})^2} = \frac{\alpha}{1-e^{-y}} - F_\gamma(y)\frac{e^{-y}}{1-e^{-y}} \\ &\geq \frac{\alpha - e^{-y}}{1-e^{-y}} = \alpha + \frac{\alpha - 1}{e^y - 1} \geq \alpha.\end{aligned}$$

Furthermore, (3) follows because α and β are determined by σ_0:

$$\alpha = \frac{e^{-\sigma_0}}{1-\sigma_0}, \quad \theta = 1 - \frac{1}{\alpha}, \text{ and } \quad \beta = \frac{1}{3\sigma_0 - \theta}.$$

By plugging in these dependencies we obtain the expression

$$\alpha(1-\beta\sigma_0)-\beta = \alpha-\beta(\alpha\sigma_0+1) = \frac{\mathrm{e}^{-\sigma_0}}{1-\sigma_0} - \frac{1}{3\sigma_0-1+\frac{1}{\frac{\mathrm{e}^{-\sigma_0}}{1-\sigma_0}}}\left(\frac{\mathrm{e}^{-\sigma_0}}{1-\sigma_0}\sigma_0+1\right),$$

which has a positive value if $\sigma_0 \in (\frac{2}{3}, 1)$. ◀

Proof of Lemma 13. Our analysis follows the basic approach from [8], but distinguishes between the cycle C and the remaining part R of the sampled solution T to leverage the lower bound on the the cost of the cycle, which we get by running the simple algorithm from Section 2 in parallel. Another difference to [8] is that T is not a single tree, but consists of k distinct sampled trees, which requires some additional notation.

Let $\mathcal{T} = \mathcal{T}_1 \times \ldots \times \mathcal{T}_k$ denote the set of all possible outcomes obtained by sampling the k trees as described in Section 2. One can see that the probability of a fixed outcome $\pi = (T_1, \ldots, T_k) \in \mathcal{T}$ is $\mu(\pi) = \prod_{i=1}^{k} \mu_i(T_i)$ and that $\sum_{\pi\in\mathcal{T}} \mu(\pi) = 1$.

Recall that technically, the subgraphs C, T as well as the layers E_i depend on the combination of sampled trees π. We therefore write, e.g., C_π to refer to the concrete cycle C that arises from sampling the trees T_i in $\pi = (T_1, \ldots, T_k)$. Now

$$\begin{aligned}
\mathbb{E}\big[c(T')+c(z_\gamma)\big] &\le \mathbb{E}\big[c(C)\big] + c(R') + \mathbb{E}\big[c(z_\gamma)\big] \\
&= \sum_{\pi\in\mathcal{T}} \mu(\pi)\left(c(C_\pi) + \mathbb{E}_\gamma\big[c(R'_\pi)\big] + \sum_{i=1}^{\ell_\pi} F_\gamma(\eta_j)\,(1-2\beta\eta_j)\,c(E_{i,\pi})\right) \\
&= \sum_{\pi\in\mathcal{T}} \mu(\pi)\left(c(C_\pi) + \sum_{i=1}^{\ell_\pi} \underbrace{\Pr[\gamma\le\eta_i]\,(2-2\beta\eta_j)}_{=:g(\eta_i)}\,c(E_{i,\pi})\right) \\
&\le \sum_{\pi\in\mathcal{T}} \mu(\pi)\Big(c(C_\pi) + g(\sigma_0)c(R_\pi)\Big) \qquad (4)\\
&= \sum_{\pi\in\mathcal{T}} \mu(\pi)\Big(c(C_\pi) + g(\sigma_0)\big(c(T'_\pi)-c(C_\pi)\big)\Big) \\
&= \sum_{\pi\in\mathcal{T}} \mu(\pi)\Big(g(\sigma_0)c(T'_\pi) - \big(f(\sigma_0)-1\big)c(C_\pi)\Big) \\
&\le g(\sigma_0)c(x) - \big(g(\sigma_0)-1\big)c(\hat{C}) = 2\big(1-\beta\sigma_0\big)c(x) - \big(1-2\beta\sigma_0\big)c(\hat{C}).
\end{aligned}$$

In (4) we used the fact from Lemma 14 that $g(\eta_i)$ is maximized at $\eta_i = \sigma_0$ and that the layers $E_{i,\pi}$ partition the edge set of R_π. Now recall that we may assume that $c(\hat{C}) \ge (\alpha-\hat{\alpha})\mathrm{opt}_{PCOTSP} \ge (\alpha-\hat{\alpha})c(x)$, which yields a bound of

$$\begin{aligned}
\mathbb{E}\big[c(T')+c(z_\gamma)\big] &\le 2\big(1-\beta\sigma_0\big)c(x) - \big(1-2\beta\sigma_0\big)c(\hat{C}) \\
&\le c(x)\Big(2-2\beta\sigma_0-(1-2\beta\sigma_0)(\alpha-\hat{\alpha})\Big) \\
&= c(x)\Big(2-2\beta\sigma_0-\alpha+\hat{\alpha}+2\alpha\beta\sigma_0-2\hat{\alpha}\beta\sigma_0\Big) \\
&= c(x)\Big(2+\hat{\alpha}-\alpha-(2+2\hat{\alpha})\beta\sigma_0+2\alpha\beta\sigma_0\Big).
\end{aligned}$$

◀

We will next analyze the cost of F_P, in Lemma 16. The proof builds on the following theorem.

▶ **Theorem 15** (Böhm et al. [13]). *Let $X \subseteq U \subseteq V$. Furthermore, let $S \subseteq U \setminus X$ be a randomly chosen subset such that $Pr[v \notin S] \le \rho$ for each $v \in U \setminus X$. Let F_X and $F_{X\cup S}$ denote the cheapest X-rooted spanning forest and the cheapest $(X \cup S)$-rooted spanning forest of U respectively. Then $\mathbb{E}\big[c(F_{X\cup S})\big] \le \rho \cdot c(F_X)$.*

▶ **Lemma 16.** *For a fixed value of $\sigma \in [\sigma_0, 1]$, the cost of F_P is at most $\frac{e^{-\sigma}}{\sigma} \cdot c(x)$.*

Proof. We invoke Theorem 15 with $U = V_\sigma$, $X = O$ and $S = (V_\sigma - O) \cap V(T)$. Recall that our pickup forest F_P is the cheapest $V_\sigma \cap V(T)$ rooted spanning forest of V_σ and observe that $V_\sigma \cap V(T) = X \cup S$. By Lemma 10, each vertex $v \in U \setminus X$ is *not* sampled (and therefore not in S) with probability at most $e^{-\sigma}$. It remains to show that the cost of the cheapest O-rooted spanning forest of V_σ is at most $\frac{c(x)}{\sigma}$.

As we have already observed in Section 2, the cost of the cheapest O-rooted spanning forest of V_σ is equal to the cost of a minimum spanning tree in the graph obtained by contracting all vertices of O into a single vertex r, i.e., on $V'' = (V_\sigma - O) \cup \{r\}$. Furthermore, we have also observed that by applying the very same contraction to our solution (x, y), we obtain a feasible solution (x', y') to the (non-ordered) PCTSP relaxation. By splitting off all vertices in $V - V_\sigma$ and scaling up by a factor of $\frac{1}{\sigma}$, we obtain a vector x'' for which $x'(\delta(s)) \geq 2$ for all $\emptyset \neq S \subseteq V''$, i.e., a feasible point in the dominant[8] of the subtours elimination polytope of V''.

The cost of the minimum spanning tree on V'' is therefore at most $c(x'') \leq \frac{1}{\sigma} c(x)$, which concludes the proof. ◀

Equipped with this upper bound, we can now bound the *expected* cost of F_P, utilizing the density function $f_\sigma(y) = \frac{d}{dy} F_\sigma(y) = \alpha y e^y$ and integrating over the range $[\sigma_0, 1]$ from which σ is drawn:

$$\mathbb{E}\big[c(F_P)\big] = c(x) \int_{\sigma_0}^{1} f_\sigma(y) \frac{e^{-y}}{y} dy = \alpha c(x) \int_{\sigma_0}^{1} 1 dy = \alpha(1 - \sigma_0) c(x) = e^{-\sigma_0} c(x). \tag{5}$$

For the last equality we have used the definition of σ_0. We emphasize that by randomizing the choice of σ instead of flatly using $\sigma = \sigma_0$, we have gained a factor of σ_0. In a similar way, we can use Lemma 16 to compute the expected cost of z_σ.

▶ **Lemma 17.** $\mathbb{E}\big[c(z_\sigma)\big] = \alpha c(x) \left(\frac{1}{4\beta} - \sigma_0 + \beta \sigma_0^2 \right)$.

Proof. Note that when $\sigma > \frac{1}{2\beta}$, then z_σ is by definition the zero-vector. We compute

$$\begin{aligned}
\mathbb{E}\big[c(z_\sigma)\big] &= \mathbb{E}\big[\max\{0, 1 - 2\beta\sigma\} c(\chi^{F_P})\big] = c(x) \int_{\sigma_0}^{\frac{1}{2\beta}} f_\sigma(y) \frac{e^{-y}(1 - 2\beta y)}{y} dy \\
&= \alpha c(x) \int_{\sigma_0}^{\frac{1}{2\beta}} (1 - 2\beta y) dy = \alpha c(x) \left(\frac{1}{2\beta} - \sigma_0 - 2\beta \int_{\sigma_0}^{\frac{1}{2\beta}} y \, dy \right) \\
&= \alpha c(x) \left(\frac{1}{2\beta} - \sigma_0 - 2\beta \left(\frac{1}{8\beta^2} - \frac{\sigma_0^2}{2} \right) \right) = \alpha c(x) \left(\frac{1}{4\beta} - \sigma_0 + \beta \sigma_0^2 \right).
\end{aligned}$$

◀

Finally, we are able to combine the bounds on the various parts of $c(H)$, and obtain our final upper bound on the expected tour cost:

$$\begin{aligned}
\mathbb{E}\big[c(H)\big] &\leq \mathbb{E}\big[c(T')\big] + \mathbb{E}\big[c(F_P)\big] + \mathbb{E}\big[c(J)\big] \\
&= \mathbb{E}\big[c(T') + c(z_\gamma)\big] + \mathbb{E}\big[c(F_P)\big] + \mathbb{E}\big[c(z_\sigma)\big] + \beta c(x^*) \\
&\leq c(x^*) \Big(2 + \hat{\alpha} + \beta - \alpha - (2 + 2\hat{\alpha})\beta\sigma_0 + 2\alpha\beta\sigma_0 + e^{-\sigma_0} + \frac{\alpha}{4\beta} - \alpha\sigma_0 + \alpha\beta\sigma_0^2 \Big).
\end{aligned}$$

[8] It is possible to obtain a feasible point in the polytope itself by applying a sequence of splitting-off operations.

Note that the values of β and σ_0 are functions of α. We therefore can express our upper bound as $\mathbb{E}\big[c(H)\big] \leq c(x^*)f(\alpha,\hat{\alpha})$. By Lemma 9, Algorithm 1 pays at most α times the penalty incurred by (x^*, y^*). Running Algorithm 1 with θ and σ determined by the single parameter α thus yields an approximation factor of $\max(\alpha, f(\alpha,\hat{\alpha}))$. Recall that the value of $\hat{\alpha}$ is currently slightly below 1.599. For $\hat{\alpha} = 1.599$, the term $\max(\alpha, f(\alpha,\hat{\alpha}))$ is minimized at $\alpha \approx 2.096896 < 2.097$. In fact, if we set $\alpha = 2.097$, the term evaluates to 2.097.

For $\alpha = 2.097$ and $\hat{\alpha} = 1.599$, we have $\beta \approx 0.548775$ and $\sigma_0 \approx 0.781790$. Thus we have proven Theorem 3.

4 Prize-collecting Multi-Path TSP

PC-Multi-Path-TSP can be modeled as the following linear program.

$$\begin{aligned} \text{minimize} \quad & \sum_{e\in E}\sum_{i=1}^{k} c_e x_{i,e} + \sum_{v\in V-\mathcal{T}} \pi_v(1-y_v) & & \\ \text{subject to} \quad & y_v = \sum_{i=1}^{k} y_{i,v} & \forall v \in V & \qquad \text{(kLP)} \\ & (x_i, y_i) \text{ lies in the } s_i\text{-}t_i\text{-stroll relaxation} & \forall i \in [k] & \end{aligned}$$

Similar to [13], we describe two algorithms for the problem and show that an appropriate combination of the two algorithms gives an 2.41-approximation.

In one algorithm, (which we call Algorithm A), we first sample k trees using Lemma 6 and an optimal solution (x^*, y^*) of (kLP), where tree T_i connects terminals s_i, t_i. The remaining vertices are picked up in a probabilistic fashion akin to Algorithm 1, i.e., we choose a random threshold $\sigma \in [\sigma_0', 1]$ and buy a $\big(V_\sigma \cap \bigcup_{i=1}^{k} V(T_i)\big)$-rooted spanning forest F_P of V_σ. Here, σ_0' is a constant whose value will be determined later. Then we double every edge that does not, for any i, lie on the s_i-t_i path in T_i. This gives a tour H_A. Define $\Delta := \sum c(s_i, t_i)$ and $\eta := \frac{\Delta}{c(x^*)}$. Then it is easy to see that

$$c(H_A) \leq 2c(x^*) + 2\mathbb{E}\big[c(F_P)\big] - \Delta.$$

One can already see that intuitively, this yields good results whenever Δ is large.

Now we define the distribution from which σ is drawn. We choose σ s.t. $\Pr[\sigma \leq y] = F'_\sigma(y)$ where $F'_\sigma(y) = 1 - \frac{e^y(1-y)}{e^{\sigma_0'}(1-\sigma_0')}$. Note that except for the constant σ_0', this is exactly the same distribution that we used in Algorithm 1. In fact, if we define $\rho := \frac{e^{-\sigma_0'}}{1-\sigma_0'}$, we obtain $F'_\sigma(y) = 1 - \frac{\rho(1-y)}{e^{-y}}$ (compare this to $F_\sigma(y) = 1 - \frac{\alpha(1-y)}{e^{-y}}$), so any result about F_σ obtained in the previous setting carries over to F'_σ if we replace α by ρ. We remark that we use the symbol ρ instead of α' because unlike in the previous case, ρ will not be our final approximation factor.

We can thus bound the expected cost of F_P in the same way as we did for PCOTSP. This is possible because the distribution F_σ as well as the (lower bound on the) probability of a vertex $v \in V_\sigma$ being sampled at least once are the same as in Section 3.3.

First, we prove an equivalent of Lemma 16, i.e., we show that $c(F_P) = \frac{e^{-\sigma}}{\sigma}$ for any fixed value of σ, and then we compute the expected cost $\mathbb{E}\big[c(F_P)\big] = e^{-\sigma_0'}$ by integrating over the range $[\sigma_0', 1]$ from which σ is chosen as we have done in Equation (5). This gives us the following upper bound on the expected tour cost:

$$c(H_A) \leq \left(2 + 2e^{-\sigma_0'} - \eta\right) c(x^*).$$

Furthermore, by similar reasoning as in Section 3.1, we know that the expected total penalty paid by this algorithm is at most $\rho = \frac{e^{-\sigma_0'}}{1-\sigma_0'}$ times the fractional penalty cost incurred by (x^*, y^*).

Now we give a simple Algorithm B that works well when η is small. Contract all the $2k$ terminals into one mega-vertex w with penalty ∞, solve the PCTSP LP for this instance, and sample a single tree T from the solution, using Lemma 6. In the original graph, T corresponds to a $\mathcal{T}$-rooted forest of cost at most $c(x^*)$ that contains each vertex v with probability at least y_v. Now double every edge of T (obtained in the original graph), and then add the k edges $\{s_i, t_i\}$ to get a final solution H_B. It is easy to see that $c(H_B) \leq (2+\eta)c(x^*)$ and that the incurred penalty is no higher than the fractional penalty cost of (x^*, y^*).

Running both algorithms A and B and returning each solution with probability $\frac{1}{2}$, yields a tour of expected cost $\frac{c(H_A)+c(H_B)}{2} \leq (2 + e^{-\sigma_0'}) \cdot c(x^*)$ and an expected total penalty cost of at most

$$\frac{1}{2}\left(\frac{e^{-\sigma_0'}}{1-\sigma_0'} + 1\right) \cdot \sum_{v \in V - \mathcal{T}} (1 - y_v^*).$$

The approximation ratio that we get from combining both algorithms is

$$\max\left\{2 + e^{-\sigma_0'}, \frac{1}{2}\left(\frac{e^{-\sigma_0'}}{1-\sigma_0'} + 1\right)\right\},$$

which is minimized at $\sigma_0 \approx 0.892769$. This yields an approximation guarantee slightly below 2.41, proving Theorem 4. We remark that both algorithm A and algorithm B can be derandomized in the same fashion as Algorithm 1.

5 Derandomization

In this section we discuss how our algorithm can be derandomized. Note that we used randomization for the choice of the thresholds σ and γ, as well as for sampling the trees $T_1, \ldots, T_k$.

For the thresholds, we follow the approach of [8]: since the thresholds are used to prune (pick up) vertices whose y-value is below (above) the respective threshold, we may generate all possible outcomes by running our algorithm once for each pair of values $\gamma, \sigma \in \{0, 1\} \cup \{y_v \mid v \in V\}$.

The sampling of the trees $T_1, \ldots, T_k$ can be derandomized using the method of conditional expectations, as it is done in [4].

The basic idea is to iteratively fix the trees $T_i := T_i^*$ for $i = 1, \ldots, k$, while minimizing the conditional expectation

$$\mathop{\mathbb{E}}_{\sigma, \gamma, T_{i+1}, \ldots, T_k} [c(T') + c(F_p) + c(J) \mid T_1 = T_1^*, \ldots, T_i = T_i^*]$$

at each step. The expected conditional costs of T' and F_p can be computed in a similar manner as it is done in [4], whereas the expected conditional cost of J can be bounded by a linear combination of the expected conditional costs of T' and F_p, and the costs of x^* and $\hat{C}$. Thus, the conditional expectation can be computed efficiently each round.

6 Discussion

The PCOTSP, as a generalization of both PCTSP and OTSP, poses the challenges of each of the individual problems plus new challenges. The best approximation algorithms for OTSP ([13, 4]) both pick up vertices which have been left out of sampling, which imposes a

cost of $\frac{1}{e}$ over the solution. In the latest PCTSP result ([8]), the cost of parity correction is slightly below 0.6. So even if we simply add up the overheads of the two problems for parity correction and vertex pickup, the approximation factor would be close to two. But a straightforward application of the techniques from the latest results on Ordered TSP and Prize-Collecting TSP to PCOTSP actually leads to an approximation factor much higher than 2. This is because in PCOTSP the sampling probability for each vertex is lower than both PCTSP and OTSP; this makes pickup more expensive. Together with the need for parity correction for the picked up vertices, this also makes parity correction more costly than PCTSP.

It is not difficult to see that for the special cases of PCTSP, OTSP, our algorithms produce the best previously-known approximation factors for these problems. For example, when $k = 1$, PCOTSP is simply PCTSP. In this case, the cycle length over the one vertex is zero, and the output of our algorithm would be no worse than the output PCTSP algorithm of [8] on the remaining vertices. Likewise, setting all penalties to ∞ turns PCOTSP into OTSP (and thus all y values are 1). Thus all vertices that have not been sampled will be picked up (i.e., $\sigma = 1$), and no vertex is split off. This is equivalent to the algorithms of [13] and [4]. In a similar vein, setting all penalties to ∞ turns PC-Multi-Path-TSP into Multi-Path-TSP, and here our algorithm would do the same as the factor 2.367 algorithm of [13]. Their factor 2.2131 algorithm, however, does not directly carry over to our setting. The issue is that in the prize collecting setting, we require a picking-up step which leads to additional costs exceeding the additional gains as soon as we have to sample more than one tree.

The distributions used in this article have been carefully balanced to achieve the target approximation factor; it seems unlikely that their further tuning leads to better factors. It is an intriguing open question whether the problem admits an approximation factor of 2 or below, which we believe requires improvements of at least one of PCTSP or OTSP.

References

1 Ali Ahmadi, Iman Gholami, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Mohammad Mahdavi. 2-approximation for prize-collecting steiner forest. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 669–693. SIAM, 2024. doi:10.1137/1.9781611977912.25.

2 Ali Ahmadi, Iman Gholami, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Mohammad Mahdavi. Prize-collecting steiner tree: A 1.79 approximation. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1641–1652, 2024. doi:10.1145/3618260.3649789.

3 Aaron Archer, MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Howard Karloff. Improved approximation algorithms for prize-collecting Steiner tree and TSP. *SIAM journal on computing*, 40(2):309–332, 2011. doi:10.1137/090771429.

4 Susanne Armbruster, Matthias Mnich, and Martin Nägele. A (3/2+ 1/e)-approximation algorithm for ordered TSP. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2024)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.

5 Jørgen Bang-Jensen, András Frank, and Bill Jackson. Preserving and increasing local edge-connectivity in mixed graphs. *SIAM Journal on Discrete Mathematics*, 8(2):155–178, 1995. doi:10.1137/S0036142993226983.

6 Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, 15:1–31, 2007.

7 Daniel Bienstock, Michel X Goemans, David Simchi-Levi, and David Williamson. A note on the prize collecting traveling salesman problem. *Mathematical programming*, 59(1):413–420, 1993. doi:10.1007/BF01581256.

8 Jannis Blauth, Nathan Klein, and Martin Nägele. A better-than-1.6-approximation for prize-collecting TSP. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 28–42. Springer, 2024. doi:10.1007/978-3-031-59835-7_3.

9 Jannis Blauth and Martin Nägele. An improved approximation guarantee for prize-collecting TSP. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1848–1861, 2023. doi:10.1145/3564246.3585159.

10 Hans-Joachim Böckenhauer, Juraj Hromkovič, Joachim Kneis, and Joachim Kupke. On the approximation hardness of some generalizations of TSP. In *Algorithm Theory–SWAT 2006: 10th Scandinavian Workshop on Algorithm Theory, Riga, Latvia, July 6-8, 2006. Proceedings 10*, pages 184–195. Springer, 2006. doi:10.1007/11785293_19.

11 Hans-Joachim Böckenhauer, Tobias Mömke, and Monika Steinová. Improved approximations for TSP with simple precedence constraints. *J. Discrete Algorithms*, 21:32–40, 2013. doi:10.1016/J.JDA.2013.04.002.

12 Martin Böhm, Zachary Friggstad, Tobias Mömke, and Joachim Spoerhase. Approximating TSP variants using a bridge lemma. *arXiv preprint arXiv:2405.12876*, 2024. doi:10.48550/arXiv.2405.12876.

13 Martin Böhm, Zachary Friggstad, Tobias Mömke, and Joachim Spoerhase. Approximating traveling salesman problems using a bridge lemma. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2025.

14 Moses Charikar and Balaji Raghavachari. The finite capacity dial-a-ride problem. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 458–467. IEEE, 1998. doi:10.1109/SFCS.1998.743496.

15 Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. *Operations Research Forum*, 3(1):20, 2022. doi:10.1007/S43069-021-00101-Z.

16 Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of operations research*, 153:29–46, 2007. doi:10.1007/S10479-007-0170-8.

17 Michel X Goemans. Combining approximation algorithms for the prize-collecting TSP. *arXiv preprint arXiv:0910.0553*, 2009. arXiv:0910.0553.

18 Michel X Goemans and David P Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995. doi:10.1137/S0097539793242618.

19 Inge Li Gørtz, Viswanath Nagarajan, and Ramamoorthi Ravi. Minimum makespan multi-vehicle dial-a-ride. *ACM Transactions on Algorithms (TALG)*, 11(3):1–29, 2015. doi:10.1145/2629653.

20 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, 2021. doi:10.1145/3406325.3451009.

21 Ian Post and Chaitanya Swamy. Linear programming-based approximation algorithms for multi-vehicle minimum latency problems (extended abstract). In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 512–531. SIAM, 2015. doi:10.1137/1.9781611973730.35.

22 Martin WP Savelsbergh and Marc Sol. The general pickup and delivery problem. *Transportation science*, 29(1):17–29, 1995. doi:10.1287/TRSC.29.1.17.

23 AI Serdjukov. Some extremal bypasses in graphs [in russian]. *Upravlyaemye Sistemy*, 17(89):76–79, 1978.

24 Vera Traub and Jens Vygen. *Approximation Algorithms for Traveling Salesman Problems*. Cambridge University Press, 2024. URL: https://books.google.de/books?id=o5jV0AEACAAJ.

25 Vera Traub, Jens Vygen, and Rico Zenklusen. Reducing path TSP to TSP. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 14–27, 2020. doi:10.1145/3357713.3384256.