Approximation Algorithms for the Traveling Thief Problem

Jan Eube
University of Bonn, Germany

Kelin Luo
University at Buffalo, NY, USA

Heiko Röglin
University of Bonn, Germany

Sarah Sturm
University of Bonn, Germany

Abstract

The Traveling Thief Problem (TTP) combines the Traveling Salesperson Problem with the Knapsack Problem. In this problem, a finite metric space is given, and at each location an item with some profit and weight is placed. An agent seeks to collect a subset of the items. To do so, the agent must decide which items to collect and to determine a cyclic tour visiting the corresponding locations. While collecting an item yields its profit as a reward, the agent's speed decreases as more weight is picked up. The problem involves two competing objectives: maximizing the total profit of the collected items and minimizing the travel time of the tour.

While many heuristics and exact algorithms (with a non-polynomial running time) have been developed, no approximation algorithms are known for any variant of the TTP. We aim at computing an (α_1, α_2)-approximate Pareto set that, for every solution, contains another solution collecting at least a $\frac{1}{\alpha_1}$ fraction of its profit while requiring at most α_2 times its travel time. Our main result is an algorithm that calculates a $(9+\epsilon, 9+\epsilon)$-approximate Pareto set in polynomial time.

We also consider the setting in which the set of items to be collected is given in advance, so that the agent only has to compute a tour through the corresponding locations that minimizes the total travel time. This is the so-called Weighted TSP. For this setting, we present a $(2e+\epsilon)$-approximation algorithm.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases Traveling Thief Problem, Traveling Salesperson Problem, Knapsack Problem, Approximation Algorithms, Bi-objective optimization

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.130

Related Version *Full Version*: `https://arxiv.org/abs/2607.05164` [12]

Funding This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 390685813; 459420781 and by the Lamarr Institute for Machine Learning and Artificial Intelligence lamarr-institute.org.

1 Introduction

The Knapsack Problem (KP) and the Traveling Salesperson Problem (TSP) are well-studied combinatorial optimization problems that have applications in many different fields. Both problems are known to be NP-hard. In order to deal with them, one either needs to accept a non-polynomial worst-case running time or rely on heuristics or approximation algorithms. While both problems are already challenging on their own, many real-world applications involve the combination of multiple NP-hard optimization problems. These combinations are often referred to as *multi-component optimization problems*. In particular, there is a

sequence of papers on a combination of the KP and the TSP called the *Traveling Thief Problem (TTP)* (e.g., [5, 17, 18, 14, 16]). Following its introduction, there were also multiple competitions that aimed at solving variants of this problem[1].

As in the TSP, also in the TTP one is given a set of locations that have to be visited by an agent (the thief) in a cyclic tour. Additionally, there are items placed at the locations and the agent has to decide which items to pick up. Each item has a profit and a weight and the agent gets slower when it picks up additional weight. The inverse speed is modeled as an increasing function, and the time needed to traverse an edge is obtained by multiplying the current function value by the edge length. There also exists a weight threshold limiting the total weight of the items that the agent is allowed to pick up. Most models in the literature consider the special case in which the agent starts with an initial speed $v_{\max}$ that decreases proportionally to the weight that has already been picked up until a minimum speed $v_{\min}$ is reached when the maximum allowed weight has been picked up. The TTP can be used to model collection and delivery settings in which routing decisions are coupled with the selection of items to be transported.

A solution of the TTP consists of a tour as well as a packing plan that determines which items are picked up in which city. One aims at both maximizing the profit of the collected items and minimizing the time necessary to traverse the tour. Given that these two objectives oppose each other, one usually searches for a trade-off between them. In the literature, often a variant is considered where a scaled version of the travel time is subtracted from the total profit of the collected items, and one aims at maximizing this difference [5, 17, 18, 16].

Alternatively, one can formulate the TTP as a bi-objective optimization problem in which the goal is to find the set of Pareto-optimal solutions or a subset thereof that contains a good solution for every possible linear combination of the two objective values [3, 7]. This viewpoint has attracted increasing attention in the evolutionary computation community. In particular, Yafrani et al. [21] highlight the connection between the bi-objective TTP and single-objective TTP, and Wu et al. [19] develop an evolutionary algorithm augmented with a dynamic programming subroutine. There is also a variant of the bi-objective TTP in which the profit of an item decreases while it is being carried [5]. Since both maximizing the profit and minimizing the travel time are NP-hard, it is already NP-hard to decide whether there exists a solution in which the difference of the collected profit and the travel time is non-negative. Consequently, one cannot expect to obtain any polynomial time approximation algorithm for this objective function, unless $\mathsf{P} = \mathsf{NP}$. For this reason, we focus in this work on approximating a TTP variant that is strongly related to the bi-objective TTP.

The current literature on approximation algorithms for the TTP is rather sparse. There exists a PTAS for the case that the tour of the agent is already fixed and one only needs to determine the packing plan [15]. For the so-called *Weighted TSP* in which the packing plan is already fixed and one only needs to determine the tour, Bossek et al. [6] provide a 3.59-approximation algorithm if the cost of traversing an edge grows linearly with the weight that has been collected. For more general cost functions, Eube et al. [11] show that the Weighted TSP is polynomially solvable if the metric is a line metric. They also show that the problem is NP-hard even on metrics induced by star-graphs and provide an 8-approximation for this case. To the authors' knowledge, there exist no approximation results if one needs to determine both the tour and the packing plan, prior to this work.

While our current theoretical knowledge regarding the TTP is rather limited, there exists a rich body of literature for the broader field of vehicle routing problems. In the classic orienteering problem, one is given an undirected graph $G = (V, E)$ with metric edge weights and node profits, a length limit, as well as a start node s and end node t. The objective is to

[1] `https://cs.adelaide.edu.au/~optlog/research/combinatorial.php`

find a path from s to t of length at most the given limit that maximizes the profit. There exists a $(2+\epsilon)$-approximation algorithm for this problem in general metrics [9]. There exist many variations of this problem. For example, Xu et al. give a constant-factor approximation for the team orienteering problem, in which multiple vehicles may be used to visit the nodes [20]. The capacitated orienteering problem (Cop) is probably the variant most closely related to the TTP [4]. In this problem, nodes also have weights that are independent of the profits, and an additional side constraint requires the total weight of the visited nodes to remain below a given threshold. If we consider the TTP with a constant speed function, the problem basically becomes the Cop problem. Bock and Sanità give a $(3+\epsilon)$-approximation for this problem [4], which we will use in our approximation algorithm for the TTP.

Also relevant to our algorithms is a variant of the TSP known as Quota TSP (Qtsp) [2]. In this problem, nodes are weighted and the objective is to find a tour of minimum length whose total weight meets a given lower bound k. If all node weights are one, this is the problem to find a tour of minimum length that visits at least k nodes. This problem is known as the k-TSP [1], and there exists a $(2+\epsilon)$-approximation algorithm for it [1].

2 Our Model

We consider a variant of the TTP that differs slightly from the existing literature. Most changes are introduced to simplify writing and do not functionally change the problem. We will discuss these changes after introducing our model.

We are given a node set V with $|V| = n$, a root $r \in V$ and a distance metric $d : V^2 \to \mathbb{R}_{\geq 0}$. At every node $v \in V \setminus \{r\}$, an item with a known weight and profit is placed. The weights of the items are given by a function $w : V \to \mathbb{N}_0$ and the profits by $p : V \to \mathbb{N}$. We assume that no item is placed at r itself, which is why we set $w(r) = p(r) = 0$. For simplicity, we also assume that the distance between r and its closest location is exactly 1, which can be ensured via scaling, unless another location is placed at the same position as r. We can model the situation that multiple items are placed at the same position by creating multiple nodes which are at distance 0 to each other.

There is an agent starting at r who wants to collect the items. To collect an item, the agent must travel to its corresponding location, and we assume that the agent visits only those locations at which items are picked up. Thus, no separate packing plan is needed. In the end, the agent has to return to the initial location r. Collecting an item yields its profit as a reward, but requires the agent to carry the item for the remainder of the tour. As the agent accumulates weight, their speed decreases. More formally, let $f : \mathbb{N}_0 \to \mathbb{R}_{\geq 0}$ be a non-decreasing function modeling the inverse speed. When the agent carries weight $W \in \mathbb{N}_0$, then traversing an edge between two vertices $u, v \in V$ requires time $f(W) \cdot d(u, v)$. The agent needs to follow a tour $\pi = (\pi_1, \ldots, \pi_{m+1})$ with $m \in [n]$ such that $\pi_1 = \pi_{m+1} = r$ and every node in $V \setminus \{r\}$ appears at most once on this tour. The entire profit collected on this tour is equal to $p(\pi) = \sum_{i=2}^{m} p(\pi_i)$. After visiting the i-th location on this tour (for an arbitrary $i \in [m]$), the agent has collected weight:

$$W_i^\pi = \sum_{j=2}^{i} w(\pi_j).$$

There is a maximum capacity $w_{\max}$ limiting the total weight that the agent can carry, and we require $w(\pi) := \sum_{i=2}^{m} w(\pi_i) \leq w_{\max}$. Without loss of generality we may assume that $f(W) = \infty$ if $W > w_{\max}$. The travel time of the agent to complete such a tour is equal to:

$$\text{cost}(\pi) = \sum_{i=1}^{m} d(\pi_i, \pi_{i+1}) f(W_i^\pi).$$

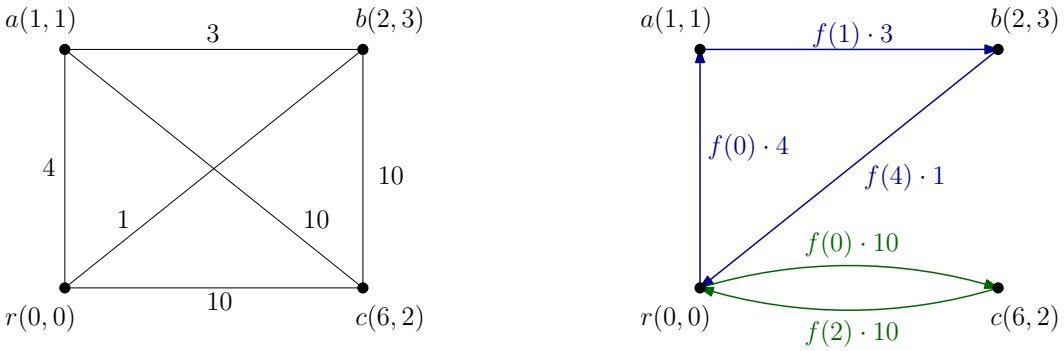

Figure 1 The left figure depicts a TTP instance. The edges are labeled with the respective distances and every node v is followed by the pair $(p(v), w(v))$. We consider the function $f(w) = w+1$ and $w_{\max} = 6$. The right figure depicts the tours (r, a, b, r) in blue and (r, c, r) in green. It holds that $p(r, a, b, r) = 3$ and $\text{cost}(r, a, b, r) = 1 \cdot 4 + 2 \cdot 3 + 5 \cdot 1 = 15$. At the same time, $p(r, c, r) = 6$ and $\text{cost}(r, c, r) = 1 \cdot 10 + 3 \cdot 10 = 40$.

We introduce the *Constrained Traveling Thief Problem* (Cttp). Similar to the Quota TSP, we assume that we are given a desired profit $\mathcal{P}$ and then aim at finding a tour that collects profit at least $\mathcal{P}$ while minimizing the travel time. Unfortunately, it is already NP-hard to decide whether there exists any tour that collects profit at least $\mathcal{P}$, as this would require to determine if there exists a subset of the locations whose entire weight is at most $w_{\max}$ and whose profit is at least $\mathcal{P}$. This is exactly the knapsack problem. To circumvent this, we design bicriteria approximation algorithms. For two constants $\alpha_1, \alpha_2 \geq 1$, a bicriteria (α_1, α_2)-approximation algorithm for Cttp calculates a tour π such that $\alpha_1 \cdot p(\pi) \geq \mathcal{P}$ and $\text{cost}(\pi) \leq \alpha_2 \cdot \text{cost}(\pi^*)$, where π^* denotes a tour with minimum travel time among all tours that collect profit at least $\mathcal{P}$. Figure 1 depicts an example Cttp instance.

While the Cttp problem has not been studied before, it is strongly related with the *Bi-objective Traveling Thief Problem* (Bttp) [3, 7]. In Bttp one aims at calculating a Pareto set with respect to the profit and the travel time of the tours. We argue in Appendix B.1 that one can use a bicriteria (α_1, α_2)-approximation algorithm for Cttp to calculate an $((1+\epsilon) \cdot \alpha_1, \alpha_2)$-approximate Pareto set for Bttp.

Most papers on the TTP assume that every location has to be visited, even if the agent does not pick up any item there. The variant in which only a subset of the nodes needs to be visited is more in line with Orienteering and related problems. Furthermore, forcing the agent to visit locations from which no item is collected seems somewhat artificial. In Appendix B.2 we argue that our results also carry over to the case that every location needs to be visited. Our model further assumes that the agent's starting position is fixed in advance and that no item is located at the corresponding node. If the initial position is not given, we can simply evaluate all n choices of r and choose the best one. We can also explicitly deal with the situation that items are placed at this starting position by modifying r and $w_{\max}$. The details of how our algorithms need to be modified are described in Appendix B.3 and B.4.

Most papers that deal with TTP also require that f corresponds to a linear speed decrease. We consider the more general setting, where f can be an arbitrary non-decreasing function.

We also consider a simplification of the Cttp. In the *Weighted TSP* (Wtsp), the packing plan is already fixed. Then we can remove a vertex from V if the respective item does not get collected. The problem becomes to find a tour π that visits every vertex and minimizes $\text{cost}(\pi)$. Neither the profit function p, nor the maximum weight $w_{\max}$ are still necessary.

3 Our Results

In Section 4 we prove the following result.

▶ **Theorem 1.** *For any $\epsilon > 0$, one can calculate a $(2e + \epsilon)$-approximation for* Wtsp *in polynomial time if all weights are polynomially bounded integers.*

The restriction to instances with polynomially bounded integer weights stems from the $(2 + \epsilon)$-approximation algorithm for the k-TSP, which we use as a subroutine in our algorithm [1]. This algorithm is stated for the case that all nodes have unit weight, but it can easily be extended to polynomially bounded integer weights by replacing each node with multiple copies. Moreover, we have examined the algorithm from [1] carefully and are confident that it also applies, without substantial modifications, to the case of general integer weights. If this is indeed the case, then Theorem 1 also extends to arbitrary integer weights.

In Section 5 we then present our main result, which is a constant bicriteria approximation algorithm for the Constrained Traveling Thief problem (Cttp).

▶ **Theorem 2.** *For any $\epsilon > 0$ there exists a polynomial time bicriteria $(9 + \epsilon, 9 + \epsilon)$-approximation algorithm for* Cttp.

4 Algorithm for a Given Packing Plan

For simplicity, we only describe an $(8 + \epsilon)$-approximation algorithm for Wtsp in this section. In Appendix A we explain how the approximation factor can be improved to $(2e + \epsilon)$.

Our algorithm for Wtsp follows the general structure of the doubling algorithm presented by Epstein et al. [10]. We consider the segments of tours that the agent travels after picking up at least some specific weight.

▶ **Definition 3.** *For a TSP-tour π we define $D^\pi(W)$ to be the distance the agent travels after picking up weight at least $W \in \mathbb{N}_0$, i.e., let i be the smallest index such that $W_i^\pi \geq W$ then we define:*

$$D^\pi(W) = \sum_{j=i}^{n} d(\pi_j, \pi_{j+1}).$$

Intuitively, if for every possible weight the distance that the agent travels after picking up that weight is bounded by a constant multiple of the corresponding distance in an optimal solution, then the total travel cost is bounded by the same constant multiple of the optimum value. Indeed, this can be shown by adapting the arguments from [10].

▶ **Lemma 4.** *Let π be a* Wtsp *tour and let π^* be the optimum* Wtsp *tour. Let c be an arbitrary constant. If it holds for any weight W that $D^\pi(W) \leq c \cdot D^{\pi^*}(W)$ then $cost(\pi) \leq c \cdot cost(\pi^*)$.*

Proof. For simplicity we define $f(-1) = 0$. For any tour π we can express $\text{cost}(\pi)$ as follows:

$$\text{cost}(\pi) = \sum_{W=0}^{w(V)} D^{\pi^*}(W) \cdot (f(W) - f(W-1))$$

Therefore we obtain:

$$\begin{aligned}\text{cost}(\pi) &= \sum_{W=0}^{w(V)} D^{\pi}(W) \cdot (f(W) - f(W-1)) \\ &\leq \sum_{W=0}^{w(V)} c \cdot D^{\pi^*}(W) \cdot (f(W) - f(W-1)) \\ &= c \cdot \text{cost}(\pi^*)\end{aligned}$$

◀

To find a tour π that fulfills the requirement of this lemma, we basically need to obtain for a given weight $\mathcal{W}$ a subtour ψ whose nodes have a total weight of at least $\mathcal{W}$ and whose length is bounded by $c \cdot D^{\pi^*}(w(V) - \mathcal{W})$ for a constant c. If we append ψ at the end of π and do not pick up the items placed on ψ during any previous subtour, we guarantee that the agent has picked up weight at most $w(V) - \mathcal{W}$ before starting with ψ. Thus, the agent does not travel much more distance with this weight than in the optimum tour. To achieve this, we use an approximation algorithm for the so-called *Quota TSP* [2].

▶ **Definition 5.** *Given a metric (V, d), a root $r \in V$, a weight function $w : V \to \mathbb{N}$ and a number $W^* \in \mathbb{N}$. The Quota TSP (QTSP) asks for a sequence $\psi = \psi_1, ..., \psi_l, \psi_{l+1}$ of vertices in V with $\psi_1 = \psi_{l+1} = r$ such that the total weight of the vertices on this sequence is at least W^* and the length of the cycle visiting all the nodes in the given order is minimized. More formally we want to find a ψ such that:*

- $\sum_{i=1}^{l} w(\psi_i) \geq W^*$
- $\ell(\psi) := \sum_{i=1}^{l} d(\psi_i, \psi_{i+1})$ *gets minimized.*

We use a subroutine for QTSP that is based on an algorithm due to Chaudhuri et al. [8]. They proved that in the special case where every vertex has unit weight (so that one simply counts the number of vertices) one can, for any given k and root r, compute in polynomial time a tree that contains r, spans at least k vertices, and has total edge length at most $(1 + \epsilon)$ times the length of the shortest path ending at r that visits at least k vertices. While Chaudhuri et al. do not address the case in which vertices have integer weights, their arguments carry over to this generalized case. Thus, one can also calculate for a desired weight W^* a tree containing r that spans weight at least W^* with length bounded by $(1 + \epsilon)$ times the length of any path ending in r that picks up weight at least W^*. Since this tree can be transformed into a tour (and thus a valid QTSP solution) by doubling all its edges and then skipping vertices that are visited repeatedly we obtain the following theorem.

▶ **Theorem 6.** *For any $\epsilon > 0$, there exists a polynomial-time algorithm that given a root r and a constant W^* finds a QTSP tour starting and ending in r whose length is upper bounded by $(2 + \epsilon)$ times the length of any path ending in r with total weight at least W^*.*

The fact that we compare the quality of the tour against a path ending in r is relevant because such paths include the subpaths of the optimum tour after a specific weight has been picked up and thus are directly related to the values $D^{\pi^*}(w(V) - \mathcal{W})$ for some given weight $\mathcal{W}$. In our algorithm, we use a subroutine $\text{QTSP}(v, (R, d), W)$ that returns a QTSP solution within this cost bound with root v and weight parameter W that only visits nodes in $R \subseteq V$ (one may note that d restricted to R is again a metric). We denote the length of this tour as $\ell(\text{QTSP}(v, (R, d), W))$. We define $\ell(\text{QTSP}(v, (R, d), W)) = \infty$ if there exists no feasible QTSP tour because W is larger than the sum of the weights of the items in R.

The idea is now to find a sequence of subtours of increasing lengths (corresponding to increasing weights) and letting the agent travel over them in reversed order, i.e., starting with the longest tour and ending with the shortest one. To avoid that multiple subtours contain

the same vertex, we maintain a set R of vertices that are not contained in any tour yet and restrict the larger tours to only these vertices. We are able to ensure that for every weight $\mathcal{W}$ there exists a subtour such that the agent does not pick up weight $\mathcal{W}$ before starting with this tour and the length of it as well as all subsequent tours is upper bounded by a constant times $D^{\pi^*}(\mathcal{W})$. This results in Algorithm 1.

Algorithm 1 An $(8+\epsilon)$-approximation for WTSP.

$R = V$, $s = 0$
while $R \neq \{r\}$ **do**
  $s = s + 1$
  Find w_s s.t. $\ell(\text{QTSP}(r, (R, d), w_s)) \leq 2^s < \ell(\text{QTSP}(r, (R, d), w_s + 1))$ (binary search)
  $\psi_s = \text{QTSP}(r, (R, d), w_s)$
  Remove all vertices in ψ_s (except r) from R
Initialize π as a sequence only containing r
for $i = s, \dots 1$ **do**
  if $\psi_i \neq (r)$ **then**
    Append ψ_i to π (ignoring the vertex r)
return: π

To analyze the approximation ratio of this algorithm, we will often be interested in the total weight picked up in the i smallest tours for a given $i \in [s]$. We define:

$$W_i^{\psi} := \sum_{j=1}^{i} w_j.$$

For any weight $\mathcal{W}$, the path that the agent travels after picking up weight at least $\mathcal{W}$ ends in r and contains vertices with weight at least $w(V) - \mathcal{W} + 1$. Given that the QTSP subroutine compares against the lengths of such paths, we can lower bound the value $D^{\pi^*}(\mathcal{W})$ as follows.

▶ **Lemma 7.** *For any $\mathcal{W}$ and $i \in [s]$ with $w(V) - \mathcal{W} \geq W_i^{\psi}$ it holds that $2^i \leq (2+\epsilon) D^{\pi^*}(\mathcal{W})$.*

Proof. Let i be an index such that $2^i > (2+\epsilon) D^{\pi^*}(\mathcal{W})$. We will show that $W_i^{\psi} > w(V) - \mathcal{W}$. Let l be the index after which the agent has picked up weight at least $\mathcal{W}$ on the optimum tour π^*. Then the path $\pi_l^*, \pi_{l+1}^*, \dots, \pi_n^*, \pi_1^*$ has exactly length $D^{\pi^*}(\mathcal{W})$. The total weight of the vertices on this path is at least $w(V) - \mathcal{W} + 1$, given that the total weight of the previous vertices is strictly smaller than $\mathcal{W}$ and thus upper bounded by $\mathcal{W} - 1$.

Let R be the set of vertices left in the graph at the beginning of the iteration of Algorithm 1 in which ψ_i gets calculated. Let ψ^* be the path that gets created by removing all vertices in $V \setminus R$ from $\pi_l^*, \pi_{l+1}^*, \dots, \pi_n^*, \pi_1^*$. Since the total weight of items in $V \setminus R$ is at most W_{i-1}^{ψ}, the weight of the vertices on this path is at least $w(V) - \mathcal{W} - W_{i-1}^{\psi} + 1$. Additionally by the triangle inequality we know that $\ell(\psi^*) \leq D^{\pi^*}(\mathcal{W})$.

For any weight W, the subroutine $\text{QTSP}(r, (R, d), W)$ returns a tour with length at most $(2+\epsilon)$ times the length of the shortest path ending in r that picks up at least the desired weight W. Thus, we know that for any $W \leq w(V) - \mathcal{W} - W_{i-1}^{\psi} + 1$ the length of the resulting tour is bounded by $(2+\epsilon)\ell(\psi^*) \leq (2+\epsilon) D^{\pi^*}(\mathcal{W}) < 2^i$ and the algorithm chooses $w_i \geq w(V) - \mathcal{W} - W_{i-1}^{\psi} + 1$. This clearly implies that $W_i^{\psi} = w_i + W_{i-1}^{\psi} \geq w(V) - \mathcal{W} + 1$ which proves the lemma. ◀

At the same time, we can bound $D^\pi(\mathcal{W})$ by the length of the $i+1$ shortest subtours for a given index $i \in [s]$ if the total weight W^ψ_{i+1} of these tours is larger than $w(V) - \mathcal{W}$.

▶ **Lemma 8.** *For any $\mathcal{W} \in \mathbb{N}_0$ and any $i \in [s]$ with $w(V) - \mathcal{W} \leq W^\psi_{i+1} - 1$ it holds that $D^\pi(\mathcal{W}) \leq 2^{i+2}$*

Proof. One may note that the entirety of vertices in the tours $\psi_{i+1}, \dots, \psi_1$ have a total weight of at least W^ψ_{i+1}. Thus, the weight that π picked up before visiting any of these vertices is bounded by $w(V) - (W^\psi_{i+1}) \leq \mathcal{W} - 1$. As a result $D^\pi(\mathcal{W})$ is upper bounded by the remaining length of π after the first vertex (except r) on one of these tours is visited. Given that we remove a vertex from R if it appears in a tour, the tours $\psi_{i+2}, \dots, \psi_n$ do not contain any vertices in $\psi_{i+1}, \dots, \psi_1$. Thus, we can bound $D^\pi(\mathcal{W})$ by the length of the combined subtours $\psi_{i+1}, \dots, \psi_1$:

$$D^\pi(\mathcal{W}) \leq \sum_{j=1}^{i+1} \ell(\psi_j) \leq \sum_{j=1}^{i+1} 2^j \leq 2^{i+2}$$ ◀

By inserting the previous two lemmas into Lemma 4 we obtain.

▶ **Theorem 9.** *If all weights are polynomially bounded integers, Algorithm 1 calculates an $(8+4\epsilon)$-approximation of the optimum WTSP tour in polynomial time.*

Proof. First, we prove the approximation factor. By Lemma 4, it is sufficient to show that for any $\mathcal{W} \in [w(V)]$ it holds that $D^\pi(\mathcal{W}) \leq (8+4\epsilon)D^{\pi^*}(\mathcal{W})$. Let $\mathcal{W}$ be chosen arbitrarily. Then there exists an index $i \in [s]$ such that $W^\psi_i \leq w(V) - \mathcal{W} \leq W^\psi_{i+1} - 1$. By Lemma 7 and 8 we obtain:

$$D^\pi(\mathcal{W}) \leq 2^{i+2} \leq 4(2+\epsilon)D^{\pi^*}(\mathcal{W}) = (8+4\epsilon)D^{\pi^*}(\mathcal{W}).$$

To bound the running time, let u, v be the two vertices maximizing the pairwise distance $d(u,v)$. Then the length of any tour returned by the QTSP algorithm has at most length $n \cdot d(u,v)$. As a result, we can bound the final value of s and thus also the iterations of the while loop by $\lceil \log(n \cdot d(u,v)) \rceil$. Given that our instance requires at least $\Omega(\log(d(u,v)))$ many bits to encode $d(u,v)$, this is bounded polynomially of the input size.

Given that we assumed that the weights are integers we also know that we can find for any $i \in [s]$ the value w_i by $O(\log(w(V)))$ uses of the algorithm QTSP which is again polynomial. Also the length of all other steps of the algorithm clearly only require polynomial time. Thus, the theorem holds. ◀

5 A Bicriteria Approximation for Constrained TTP

In this section we develop a bicriteria approximation algorithm for the Constrained Traveling Thief Problem (CTTP). For this, we need another method to bound the cost of a tour against the cost of an optimum solution. Let us assume w.l.o.g. that $f(0) = 1$ (which can be ensured via scaling). We subdivide the range of potential weights $0, \dots, w_{\max}$ into intervals in which the speed of the agent does not differ by more than a factor of two. To be more precise let $s = \lceil \log(f(w_{\max}) \rceil - 1$. Then for every $i \in [s]$ we define a value T_i such that $f(T_i) < 2^i \leq f(T_i + 1)$ and for $i = 0$ we set $T_0 = 0$ and for $i = s+1$ we set $T_{s+1} = w_{\max}$. Then we can lower bound the cost of the optimum solution as follows.

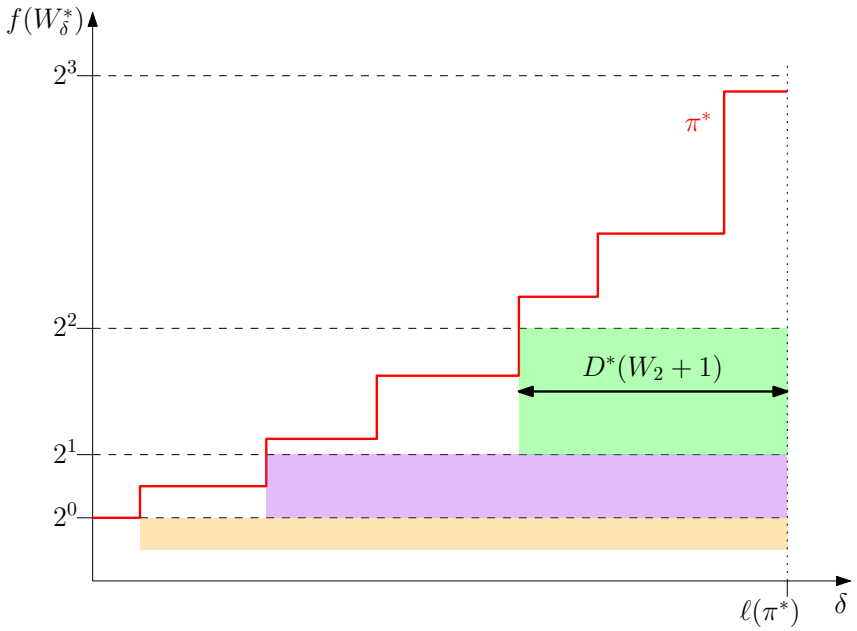

Figure 2 A sketch depicting how the cost of the optimum solution can be lower bounded. For a distance δ we define W^*_δ to be the weight that the agent has picked up on the optimum tour π^* after traveling distance δ. The red curve depicts the value of the cost function f over the course of π^* and it is easy to verify that the cost of the tour is equal to the area below this curve. In Lemma 10 we lower bound this by the sum of the areas of the colored rectangles.

▶ **Lemma 10.** *Let π^* be the optimum tour. It holds that*

$$cost(\pi^*) \geq \sum_{i=0}^{s} 2^{i-1} \cdot D^{\pi^*}(T_i + 1).$$

Proof. Figure 2 sketches the idea behind this proof. We consider an alternative cost function g with $g(w) = 2^i$ for any $w \in \{T_i + 1, \dots, T_{i+1}\}$ and denote the cost of tour π^* if we use g instead of f as $cost'(\pi^*)$. It is easy to observe that for any $w \in [w_{\max}] \cup \{0\}$ it holds that $f(w) \geq g(w)$ and thus $cost(\pi^*) \geq cost'(\pi^*)$. For any $i \in \{0, \dots, s-1\}$ we denote the length of the subpath on which the agent has picked up weight at least $T_i + 1$ and less than $T_{i+1} + 1$ by ℓ_i. One might note that ℓ_i might be 0 if there exists a vertex where the agent arrives with weight smaller $T_i + 1$ and leaves with weight at least $T_{i+1} + 1$. For $i = s$ we define ℓ_s as the length of the subpath after the agent has picked up weight at least $T_s + 1$. Then

$$\begin{aligned} cost'(\pi^*) &\geq \sum_{i=0}^{s} g(T_i + 1) \cdot \ell_i \\ &= \sum_{i=0}^{s} 2^i \cdot \ell_i \\ &\geq \sum_{i=0}^{s} (2^i - 2^{i-1}) \cdot \sum_{j=i}^{s} \ell_j \\ &= \sum_{i=0}^{s} 2^{i-1} \cdot D^{\pi^*}(T_i + 1), \end{aligned}$$

where we used in the last inequality that $D^{\pi^*}(T_i+1)$ denotes exactly the length of the subtour after the agent has picked up weight at least T_i+1 which can be subdivided into the subpaths corresponding to $\ell_1,\dots,\ell_s$. ◀

Intuitively, this lemma will allow us to calculate for every interval $[T_i+1, T_{i+1}]$ a subtour on which the collected weight lies within this interval. Given that the speed of the agent only differs by 2 on this subtour the exact order of the locations visited is not too important. If for every i the length of the subtour is bounded by $D^{\pi^*}(T_i+1)$ we know that if we append these tours after each other the total travel time of the resulting tour is then also bounded by a constant times $\text{cost}(\pi^*)$.

For any $i \in [s]\cup\{0\}$ we define the index b_i such that $W^{\pi^*}_{b_i-1} \leq T_i$ and $W^{\pi^*}_{b_i} \geq T_i+1$. Or in other words the locations $\pi^*_{b_1},\dots,\pi^*_{b_s}$ are exactly those where the weight that the agent picked up moves into another interval. For technical reasons we also define $b_{s+1} = m+1$ (where m is the number of items collected by the optimum tour). We define $B = \{\pi^*_{b_i} \mid i \in [s]\cup\{0\}\}$. We present two algorithms that compute two possible CTTP tours $\pi^{(1)}$ and $\pi^{(2)}$ such that both $\text{cost}(\pi^{(1)})$ and $\text{cost}(\pi^{(2)})$ can be bounded by a constant times $\text{cost}(\pi^*)$. Additionally, we have that if $p(B)$ is not too large then $p(\pi^{(1)})$ is at least a constant fraction of $\mathcal{P}$. Otherwise, $p(\pi^{(2)})$ can be lower bounded and we obtain a bicriteria approximation by always taking the more profitable tour.

To calculate $\pi^{(1)}$, we consider for any $i \in [s]\cup\{0\}$ the path $(\pi^*_{b_i+1},\dots,\pi^*_{b_{i+1}-1}, r)$ that visits all vertices on the optimum tour appearing after $\pi^*_{b_i}$ and before $\pi^*_{b_{i+1}}$. Then the length of this path is upper bounded by $D^{\pi^*}(T_i+1)$ and the weight picked up by $T_{i+1}-T_i$. The existence of such a path will enable us to find a subtour corresponding to the weight interval $[T_i+1, T_{i+1}]$ that is not too expensive (when starting with the weight collected by the subtours corresponding to the previous intervals) and picks up items whose profit is lower bounded by a constant times $\sum_{j=b_i+1}^{b_{i+1}-1} p(\pi^*_j)$. Since every item in $V(\pi^*)$ that is not contained in B is contained in one of this described paths we will be able to use this to lower bound the values of $\pi^{(1)}$. The details are presented in Section 5.1.

To obtain the tour $\pi^{(2)}$, we can use Lemma 10 to conclude that a tour that visits the items in B in the same order as π^* and returns to r between any two items does not take too much longer than π^*. Since the items are directly visited from r, the distance that needs to be traveled to reach an item is independent from the remainder of the tour. As a result, we can use a knapsack algorithm that yields a tour whose profit is at least a constant fraction of $p(B)$. The details are presented in Section 5.2.

For both algorithms we need to have an estimate for the travel time of the optimum tour. We assume that a value $\mathcal{D}$ is given such that $\text{cost}(\pi^*) \leq \mathcal{D} \leq (1+\epsilon)\text{cost}(\pi^*)$. This can be ensured by trying out all values $(1+\epsilon)^t$ for $t \leq \lceil \max_{v,w\in V} \log_{(1+\epsilon)}(n \cdot d(v,w)\cdot f(w_{\max}))\rceil$.

Additionally, we denote for a subtour $\psi = (\psi_1,..,\psi_m,\psi_{m+1})$ where $\psi_1 = \psi_{m+1}$ and a weight W the cost of ψ when the agent collected already weight W as

$$\text{cost}(\psi, W) = \sum_{i=1}^{m} d(\psi_i,\psi_{i+1}) f(W_i^{\psi} + W).$$

This definition will help us when bounding the cost of the calculated solutions.

5.1 Obtaining the First Solution

To obtain the solution $\pi^{(1)}$ we need some results for the so called *capacitated orienteering problem* (COP) [4]. In this problem one is given a point set V, a distance metric $d: V^2 \to \mathbb{R}_{\geq 0}$, a weight function $w: V \to \mathbb{N}$, a reward function $p: V \to \mathbb{N}$, a capacity $C \in \mathbb{N}$, a length

restriction $L \in \mathbb{R}_{\geq 0}$ and a root $r \in V$. The objective is to obtain a path ψ ending in r such that the total weight of the vertices on the path is bounded by C, the length is bounded by L and the reward of the vertices on the path gets maximized. Bock and Sanità [4] showed the following theorem.

▶ **Theorem 11.** *For any constant $\epsilon > 0$, one can $(3+\epsilon)$-approximate the capacitated orienteering problem in polynomial time. If the distance metric d is Euclidean the approximation factor can be improved to $(1+\epsilon)$.*

In our algorithm we use a subroutine $\textsc{Cop}(S, C, L)$ that returns a path ending in r that only visits vertices in $S \subseteq V$ such that its weight is at most C, its length at most L and its profit is lower bounded by $\frac{1}{3+\epsilon}$ times the profit of any other path fulfilling these requirements. We directly interpret the result of the subroutine as a subtour on which the agent travels from r to the start of the path and then follows the path back to r.

If we knew the value $D^{\pi^*}(T_i + 1)$ for an $i \in [s] \cup \{0\}$ we could use this algorithm to find a "good" subtour of length at most $D^{\pi^*}(T_i + 1)$ that picks up at most weight $T_{i+1} - T_i$ which means that the collected weight stays within the interval $[T_i, T_{i+1}]$ if the agent collected at most weight T_i on the previous tours. We may "guess" the value of $D^{\pi^*}(T_i + 1)$ with accuracy $(1+\epsilon)$ by trying out all values $(1+\epsilon)^j$ for reasonable integer values j, but it is not directly possible to detect whether or not a guess is correct. For two different guesses of the length, it is possible (and actually likely) that for one guess the resulting tour produced by Cop will take less time to traverse while the tour of the other guess picks up a larger profit. The choice which tour is preferable depends on the knowledge how much the agent needs to travel within the other intervals to pick up a certain profit.

To circumvent this problem, we may use our estimate $\mathcal{D}$ of the cost of the optimum tour. If we only consider "efficient" subtours on which the ratio between the profit and the cost of the tour is lower bounded by a constant times $\frac{\mathcal{P}}{\mathcal{D}}$ then we know that the cost of the resulting tour can only exceed the cost of the optimum tour if we also pick up a constant fraction of the profit. If the cost of the optimum solution is exceeded by too much, we can simply remove some of the subtours. When considering the possible subtours for an interval, we pick the most profitable among all efficient subtours. We will be able to use Lemma 10 to ensure that then at least a constant fraction of the optimum profit is picked up (if $p(B)$ is not too large). We end up with Algorithm 2.

For any $i \in [s] \cup \{0\}$ we consider the subtour that picks up all items that are collected on π^* while the collected weight lies within $[T_i, T_{i+1}]$. If the ratio between the profit and the travel time of this tour (if one starts with weight T_i) is not too low we call the interval $[T_i, T_{i+1}]$ efficient. We provide a formal definition that also extends to the case that some items have already been collected:

▶ **Definition 12.** *For any $i \in [s] \cup \{0\}$ we define the set $A_i = \{\pi^*_j \mid T_i + 1 \leq W^{\pi^*}_{j-1} \leq T_{i+1} - w(\pi^*_j)\}$ of items that are visited on the optimum tour such that both directly before and after this visit the collected weight lies within $[T_i + 1, T_{i+1}]$.*

For any set $P \subseteq V$ we say that the interval $[T_i + 1, T_{i+1}]$ is efficient after P has been collected if $\frac{p(A_i \setminus P)}{D^{\pi^*}(T_i+1) \cdot 2^{i+1}} \geq \frac{1}{3} \cdot \frac{\mathcal{P}}{4 \cdot \mathcal{D}}$.

If an interval is efficient this guarantees us a good solution that we can compare against when calculating ψ_i:

▶ **Lemma 13.** *For any $i \in [s] \cup \{0\}$ it holds that if the interval $[T_i + 1, T_{i+1}]$ is efficient after P_i has been collected then $p(\psi_i) \geq \frac{p(A_i \setminus P_i)}{3+\epsilon}$.*

Algorithm 2 Calculation of the first tour $\pi^{(1)}$.

$P_0 = \emptyset$, $t_1 = \frac{1}{9+3\epsilon} \cdot \frac{\mathcal{P}}{(6+6\epsilon)\mathcal{D}}$
for $i = 0, \dots s$ **do**
  $\psi_i = (r)$
  for $j = 0, \dots, \lceil \log_{1+\epsilon}(\mathcal{D}/2^i) \rceil$ **do**
    $\psi' = \textsc{Cop}((V \setminus P_i, d), T_{i+1} - T_i, (1+\epsilon)^j)$
    if $\frac{p(\psi')}{cost(\psi', w(P_i))} \geq t_1$ **then**
      if $p(\psi') \geq p(\psi_i)$ **then**
        Set $\psi_i = \psi'$
  Set $P_{i+1} = P_i \cup V(\psi_i)$
Initialize $\pi^{(1)}$ as a sequence only containing r
$i = 0$
while $cost(\pi^{(1)}) \leq (6+6\epsilon)\mathcal{D}$ *and* $i \leq s$ **do**
  if $\psi_i \neq (r)$ **then**
    Append ψ_i to $\pi^{(1)}$ (ignoring the vertex r)
  $i = i + 1$
return: $\pi^{(1)}$

Proof. Let us consider the iteration in which we determine ψ_i. Let $j = \lceil \log_{(1+\epsilon)}(D^{\pi^*}(T_i+1)) \rceil$. Since $f(T_i+1) \geq 2^i$ we know that $D^{\pi^*}(T_i+1) \leq \frac{\text{cost}(\pi^*)}{2^i} \leq \frac{\mathcal{D}}{2^i}$. Thus, the algorithm will consider $\psi' = \textsc{Cop}(V \setminus P_i, T_{i+1} - T_i, (1+\epsilon)^j)$ as a possible choice of ψ_i. Let us consider the path that visits the items in $A_i \setminus P_i$ in the same order as π^* and then returns to r. Then the length of this path is upper bounded by $D^{\pi^*}(T_i+1)$ and its weight by $T_{i+1} - T_i$. The total profit of this path is $p(A_i \setminus P_i)$, which together with the fact that Cop is a $(3+\epsilon)$-approximation implies that $p(\psi') \geq \frac{p(A_i \setminus P_i)}{3+\epsilon}$.

At the same time the length of the path corresponding to ψ' is at most $(1+\epsilon)D^{\pi^*}(T_i+1)$. Since on the subtour the agent first travels from r to the start of this path and then traverses the path itself we may bound $\text{cost}(\psi', w(P_i))$ as follows:

$$\begin{aligned}
\text{cost}(\psi', w(P_i)) &\leq (1+\epsilon)D^{\pi^*}(T_i+1)f\big(w(P_i)\big) + (1+\epsilon)D^{\pi^*}(T_i+1)f\big(w(P_i) + T_{i+1} - T_i\big) \\
&\leq (1+\epsilon)D^{\pi^*}(T_i+1)(f(T_i) + f(T_{i+1})) \\
&\leq (1+\epsilon)D^{\pi^*}(T_i+1)(2^i + 2^{i+1}) \\
&= 1.5 \cdot (1+\epsilon) \cdot \big(D^{\pi^*}(T_i+1) \cdot 2^{i+1}\big) \\
&\leq \frac{3 \cdot (6+6\epsilon) \cdot \mathcal{D} \cdot p(A_i \setminus P_i)}{\mathcal{P}}.
\end{aligned}$$

Thus:

$$\frac{p(\psi')}{\text{cost}(\psi', w(P_i))} \geq \frac{p(A_i \setminus P_i)}{3+\epsilon} \cdot \frac{\mathcal{P}}{3 \cdot (6+6\epsilon) \cdot \mathcal{D} \cdot p(A_i \setminus P_i)} = t_1.$$

As a result the algorithm will choose $\psi_i = \psi'$ unless it finds another more profitable subtour and $p(\psi_i)$ is lower bounded by $p(\psi') \geq \frac{p(A_i \setminus P_i)}{3+\epsilon}$. ◀

By bounding the profit that corresponds to the inefficient intervals, we are able to prove that there still exists an efficient interval if the total profit of the already collected items is smaller than $\frac{\mathcal{P}}{9+3\epsilon}$.

▶ **Lemma 14.** *If $p(B) \leq \frac{2+\epsilon}{9+3\epsilon}\mathcal{P}$ it holds for any $i \in [s] \cup \{0\}$ that if $p(P_i) < \frac{\mathcal{P}}{9+3\epsilon}$ then there exists an $i' \geq i$ such that the interval $[T_{i'}+1, T_{i'+1}]$ is efficient after P_i has been collected.*

Proof. See Appendix C.2 ◀

Since there do not exist any intervals with indices greater than $s+1$, we can use this lemma to directly lower-bound the total profit collected by the intervals $\psi_1, \ldots, \psi_s$:

▶ **Corollary 15.** *After executing Algorithm 2, it holds that $p(P_{s+1}) \geq \frac{\mathcal{P}}{9+3\epsilon}$ if $p(B) \leq \frac{2+\epsilon}{9+3\epsilon}\mathcal{P}$.*

While this lower-bounds the total profit collected by the subtours, Algorithm 2 might not append all subtours to $\pi^{(1)}$. But, this can only happen if $\text{cost}(\pi^{(1)}) \geq (6+6\epsilon)\mathcal{D}$ and we can lower-bound the ratio between the profit and the cost of $\pi^{(1)}$.

▶ **Lemma 16.** *If $p(B) \leq \frac{2+\epsilon}{9+3\epsilon}\mathcal{P}$ it holds that $p(\pi^{(1)}) \geq \frac{1}{9+3\epsilon}\mathcal{P}$.*

Proof. We distinguish two cases. In the first one $\pi^{(1)}$ consists of all the subtours $\psi_0, \ldots, \psi_s$ appended to each other. By Corollary 15 we know that $p(P_{s+1}) \geq \frac{\mathcal{P}}{9+3\epsilon}$. Since P_{s+1} is the union of the items that are collected on the subtours $\psi_0, \ldots, \psi_s$ this implies that $p(\pi^{(1)}) = p(P_{s+1}) \geq \frac{\mathcal{P}}{9+3\epsilon}$.

In the second case, there exists an $i \in [s]$ such that the algorithm does not append ψ_i to $\pi^{(1)}$. This can only happen if $\text{cost}(\pi^{(1)}) > (6+6\epsilon)\mathcal{D}$ when the algorithm considers to append the subtour ψ_i to $\pi^{(1)}$. Then the tour $\pi^{(1)}$ consists of the subtours $\psi_0, \ldots, \psi_{i-1}$ that get traversed after each other. One can bound its travel time as follows:

$$\begin{aligned}\text{cost}(\pi^{(1)}) &\leq \sum_{j=0}^{i-1} \text{cost}\left(\psi_j, w(P_i)\right) \\ &\leq \sum_{j=0}^{i-1} \frac{p(\psi_j)}{t_1} \\ &= \frac{p(\pi^{(1)})}{t_1}.\end{aligned}$$

Thus, the profit of $\pi^{(1)}$ is lower bounded by its travel time times t_1 and since its travel time greater than $(6+6\epsilon)\mathcal{D}$ we obtain:

$$p(\pi^{(1)}) > t_1 \cdot (6+6\epsilon)\mathcal{D} = \frac{1}{9+3\epsilon} \cdot \frac{\mathcal{P}}{(6+6\epsilon)\mathcal{D}}(6+6\epsilon)\mathcal{D} = \frac{\mathcal{P}}{9+3\epsilon}.$$

Thus, the lemma is proven. ◀

Lastly, we have to prove that the cost of $\pi^{(1)}$ is not too large.

▶ **Lemma 17.** *It holds that cost$(\pi^{(1)}) \leq (9+9\epsilon)\mathcal{D}$.*

Proof. Let $i \in [s]$ be the index of the last tour that gets appended to $\pi^{(1)}$. Let $\sigma^{(1)}$ denote the tour $\pi^{(1)}$ before ψ_i gets appended. Since Algorithm 2 only continues to append subtours as long as $\text{cost}(\pi^{(1)}) \leq (6+6\epsilon)\mathcal{D}$ we know that $\text{cost}(\sigma^{(1)}) \leq (6+6\epsilon)\mathcal{D}$. We can bound the cost of $\pi^{(1)}$ as follows:

$$\text{cost}(\pi^{(1)}) \leq \text{cost}(\sigma^{(1)}) + \text{cost}(\psi_i, w(\sigma^{(1)})) \leq (6+6\epsilon)\mathcal{P} + \text{cost}(\psi_i, w(\sigma^{(1)})).$$

We know that the weight of the items picked up on the tours $\psi_1, \ldots, \psi_{i-1}$ is upper bounded by T_i and that the weight picked up tour ψ_i is upper bounded by $T_{i+1} - T_i$. Thus, at the

beginning of the subtour ψ_i the collected weight is upper bounded by T_i and at the end it is upper bounded by T_{i+1}. Additionally, we know that the length of the path after the agent collects the first item on the subtour ψ_i is upper bounded by $(1+\epsilon)^{\lceil \log_{1+\epsilon}(\mathcal{D}/2^i)\rceil} \leq (1+\epsilon)\frac{\mathcal{D}}{2^i}$. By the triangle inequality we may also bound the distance from the r to the location of the first item by this value and we obtain:

$$\begin{aligned}\text{cost}(\psi_i, w(\sigma^{(1)})) &\leq (1+\epsilon)\frac{\mathcal{D}}{2^i}f(T_i) + (1+\epsilon)\frac{\mathcal{D}}{2^i}f(T_{i+1}) \\ &\leq (1+\epsilon)\frac{\mathcal{D}}{2^i}(2^i + 2^{i+1}) \\ &\leq (3+3\epsilon)\mathcal{D}\end{aligned}$$

Thus, $\text{cost}(\pi^{(1)}) \leq (6+6\epsilon)\mathcal{D} + (3+3\epsilon)\mathcal{D} = (9+9\epsilon)\mathcal{D}$ and the lemma is proven. ◀

5.2 Obtaining the Second Solution

We describe an algorithm that calculates a tour $\pi^{(2)}$ whose profit can be lower bounded in terms of the profit of the items in $p(B)$. The tour basically consists of subtours on which the agent collects a single item and returns to r in between. The algorithm will only collect an item (after picking up a certain weight) when the cost of doing this is upper bounded by a threshold t_2 times the profit of the item. Since the agent only gets slower over time, we can define for any item $v \in V$ a weight $s(v)$ such that we are willing to pick up v if and only if the already collected weight is at most $s(v)$. Additionally, we also require that $\text{cost}((r,v,r), s(v)) \leq 2\mathcal{D}$ to prevent that a single subtour is too expensive. Our algorithm maximizes the profit of the collected items such that no item v is collected after the agent already collected items with total weight greater $s(v)$. This can be modeled as the *Multiperiod Binary Knapsack Problem* (MPBKP) which is a knapsack variant with deadlines [13]:

▶ **Definition 18.** *In the* Multiperiod Binary Knapsack Problem *(MPBKP) one is given a set of items V, a weight function $w : V \to \mathbb{N}_0$, a profit or reward function $p : V \to \mathbb{N}_0$ and a function $s : V \to \mathbb{N}_0$ that assigns a deadline to every item. The aim is to find a tuple $\sigma = (\sigma_1, \ldots, \sigma_l)$ of distinct items such that the profit $p(\sigma) = \sum_{i=1}^{l} p(\sigma_i)$ gets maximized and for any $j \in [l]$ it holds that $\sum_{i=1}^{j-1} w(\sigma_i) \leq s(\sigma_j)$.*

The definition differs from the one presented by Gao et al. [13] but it is easy to verify that both definitions are equivalent. They provided a fully polynomial $(1+\epsilon)$-approximation scheme for this problem. In our algorithm we use this scheme as a function $\text{MPBKP}(V, p, w, s)$ to calculate a $(1+\epsilon)$ approximation. After calculating the respective solution, we simply append items to the initially empty tour $\pi^{(2)}$ until $\text{cost}(\pi^{(2)})$ exceeds $6\mathcal{D}$ or no items remain. Algorithm 3 provides the pseudocode of this procedure.

Similarly to the analysis of Algorithm 2, we subdivide the items in B into two groups, efficient and inefficient items:

▶ **Definition 19.** *Let $\beta_1, \ldots, \beta_m$ be the elements of B ordered by the moment in which the agent visits the respective location in π^*. For any $j \in [m]$ let $W^*_{\beta_j}$ be the weight that the agent has collected on π^* once it reaches β_j. We define i_j as the largest index such that $W^*_{\beta_j} \geq T_{i_j} + 1$. We call β_j* efficient *if $\frac{p(\beta_j)}{D^{\pi^*}(T_{i_j}+1)\cdot 2^{i_j-1}} \cdot (9+3\epsilon) \geq \frac{\mathcal{P}}{\mathcal{D}}$.*

By the definition of B, we know for any $\beta_j \in B$ that the weight collected before and after visiting β_j cannot be in the same interval $[T_i + 1, T_{i+1}]$ for an $i \in [s] \cup \{0\}$. Because of this, the indices $i_1, \ldots, i_m$ need to be strictly increasing.

Algorithm 3 Calculation of the second tour $\pi^{(2)}$.

$t_2 = \frac{6\cdot(9+3\epsilon)\cdot\mathcal{D}}{\mathcal{P}}$
Define the function $s : V \to \mathbb{N}_0$
for $v \in V \setminus \{r\}$ **do**
  Calculate the value $s(v)$ such that
  $\text{cost}((r,v,r), s(v)) \leq \min(p(v)\cdot t_2, 2\mathcal{D}) < \text{cost}((r,v,r), s(v)+1)$
$(\sigma_1, \dots, \sigma_l) = \text{MPBKP}(V, p, w, s)$
Find minimum $l' \in [l]$ such that $\text{cost}(r, \sigma_1, \dots, \sigma_{l'}, r) \geq 6\mathcal{D}$.
If no such l' exists set $l' = l$.
$\pi^{(2)} = (r, \sigma_1, \dots, \sigma_{l'}, r)$
return: $\pi^{(2)}$

▶ **Observation 20.** *For any $j \in [m-1]$ it holds that $i_j < i_{j+1}$.*

One can show that one can pack the efficient items in B into the knapsack without violating the respective deadlines by using the order in which they are visited on the optimum tour π^*.

▶ **Lemma 21.** *For any $j \in [m]$ it holds that $s(\beta_j) \geq \sum_{j'=1}^{j-1} w(\beta_{j'})$ if β_j is efficient.*

Proof. Let $W^\beta_{j-1} = \sum_{j'=1}^{j-1} w(\beta_{j'})$. It is easy to verify that $W^\beta_{j-1} + w(\beta_j) \leq T_{i_j+1}$. Additionally by Observation 20 we know that $i_{j-1} \leq i_j - 1$ and thus $W^\beta_{j-1} \leq T_{i_j}$. Thus

$$\begin{aligned}
\text{cost}((r,\beta_j,r), W^\beta_{j-1}) &= d(r,\beta_j)\cdot f\left(W^\beta_{j-1}\right) + d(\beta_j,r)\cdot f\left(W^\beta_{j-1}+w(\beta_j)\right)\\
&\leq d(r,\beta_j)\cdot\left(f(T_{i_j}) + f(T_{i_j+1})\right)\\
&\leq d(r,\beta_j)\cdot(2^{i_j} + 2^{i_j+1})\\
&\leq 6\cdot(D^{\pi^*}(T_{i_j}+1)\cdot 2^{i_j-1})\\
&\leq 6\cdot\left(p(\beta_j)\cdot(9+3\epsilon)\cdot\frac{\mathcal{D}}{\mathcal{P}}\right)\\
&= p(\beta_j)\cdot t_2.
\end{aligned}$$

Additionally, we know that on the optimum tour the agent travels at least distance $D^{\pi^*}(W^\beta_{j-1}+\beta_j) \geq d(r,\beta_j)$ after picking up weight $W^\beta_{j-1}+\beta_j$. Thus,

$$\text{cost}((r,\beta_j,r), W^\beta_{j-1}) \leq 2f(W^\beta_{j-1}+w(\beta_j))\cdot D^{\pi^*}(W^\beta_{j-1}+w(\beta_j)) \leq 2\mathcal{D}.$$

By combining these two bounds we obtain $\text{cost}((r,\beta_j,r), W^\beta_{j-1}) \leq \min(p(\beta_j)\cdot t_2, 2\mathcal{D})$ which implies that $s(\beta_j) \geq W^\beta_{j-1}$. ◀

This implies that the efficient items in B form a feasible MPBKP solution. By bounding the total profit of the inefficient items in B, we obtain a lower bound for the profit of σ:

▶ **Lemma 22.** *If $p(B) \geq \frac{2+\epsilon}{9+3\epsilon}\mathcal{P}$ then it holds that $p(\sigma) \geq \frac{\mathcal{P}}{9+3\epsilon}$.*

Proof. Let us consider the MPBKP solution $(\beta_{j_1}, \dots, \beta_{j_{m'}})$ such that $j_1, \dots, j_{m'}$ are exactly the indices of the efficient items in B in the order in which they are visited on the optimum tour. One may verify that this is a feasible solution because for any $z \in [m']$ it holds that

$$\sum_{z'=1}^{z-1} w(\beta_{j_{z'}}) \leq \sum_{j'=1}^{j_z-1} w(\beta_{j'}) \overset{\text{Lem. 21}}{\leq} s(\beta_{j_z}).$$

It remains to lower bound the profit of this solution. To do this let I_{inef} be a set containing the indices of all inefficient items in B. Then

$$\begin{aligned}\sum_{j\in I_{inef}} p(\beta_j) &< \frac{\mathcal{P}}{(9+3\epsilon)\mathcal{D}} \sum_{j\in I_{inef}} D^{\pi^*}(T_{i_j}+1)\cdot 2^{i_j-1} \\ &\overset{\text{Obs. 20}}{\leq} \frac{\mathcal{P}}{(9+3\epsilon)\mathcal{D}} \sum_{i=0}^{s} D^{\pi^*}(T_i+1)\cdot 2^{i-1} \\ &\overset{\text{Lem. 10}}{\leq} \frac{\mathcal{P}}{(9+3\epsilon)\mathcal{D}}\text{cost}(\pi^*) \\ &\leq \frac{\mathcal{P}}{9+3\epsilon}\end{aligned}$$

Thus, the total profit of the efficient items can be lower bounded by

$$p(B) - \sum_{j\in I_{inef}} p(\beta_j) > \frac{2+\epsilon}{9+3\epsilon}\mathcal{P} - \frac{\mathcal{P}}{9+3\epsilon} = \frac{1+\epsilon}{9+3\epsilon}\mathcal{P}.$$

Since σ is a $(1+\epsilon)$-approximation of the optimum MPBKP solution, this implies that $p(\sigma) \geq \frac{\mathcal{P}}{9+3\epsilon}$. ◀

As in the proof of Lemma 16, we can show that if $\pi^{(2)}$ does not collect all items contained in σ then the profit of $\pi^{(2)}$ can be lower bounded because the ratio between its cost and profit is bounded. If $\pi^{(2)}$ collects all items in σ, we can apply the previous lemma to lower bound the profit of $\pi^{(2)}$.

▶ **Lemma 23.** *If $p(B) \geq \frac{2+\epsilon}{9+3\epsilon}\mathcal{P}$ then it holds that $p(\pi^{(2)}) \geq \frac{\mathcal{P}}{9+3\epsilon}$.*

Proof. See Appendix C.2 ◀

Lastly, we bound the travel time of the tour $\pi^{(2)}$:

▶ **Lemma 24.** *It holds that cost$(\pi^{(2)}) \leq 8\mathcal{D}$.*

Proof. Let l' be chosen such that $\pi^{(2)} = (r, \sigma_1, \ldots, \sigma_{l'}, r)$. Then it holds that $\text{cost}(r, \sigma_1, \ldots, \sigma_{l'-1}, r) \leq 6\mathcal{D}$ and we may bound $\text{cost}(\pi^{(2)})$ as follows:

$$\begin{aligned}\text{cost}(\pi^{(2)}) &\leq 6\mathcal{D} + \text{cost}\left((r, \sigma_{l'}, r), \sum_{j=1}^{l'-1} w(\sigma_j)\right) \\ &\leq 6\mathcal{D} + \text{cost}((r, \sigma_{l'}, r), s(\sigma_{l'})) \\ &\leq 6\mathcal{D} + 2\mathcal{D} \\ &= 8\mathcal{D},\end{aligned}$$

where we used in the last inequality that $s(\sigma_{l'})$ is chosen in such a way that $\text{cost}((r, \sigma_{l'}, r), s(\sigma_{l'})) \leq 2\mathcal{D}$. ◀

By executing both Algorithm 2 and Algorithm 3 and taking the tour with the larger profit, we obtain a bicriteria approximation for CTTP:

▶ **Theorem 25.** *For given $\mathcal{P} \in \mathbb{N}$ and an $\epsilon > 0$ one can compute in polynomial time a CTTP tour picking up profit at least $\frac{1}{9+\epsilon}\mathcal{P}$ whose travel cost is upper bounded by $(9+\epsilon)$ times the optimum travel cost of any tour picking up profit at least $\mathcal{P}$.*

References

1 Sanjeev Arora and George Karakostas. A 2+ϵ approximation algorithm for the k-MST problem. *Mathematical Programming*, 107(3):491–504, 2006.

2 Giorgio Ausiello, Vincenzo Bonifaci, Stefano Leonardi, and Alberto Marchetti-Spaccamela. Prize collecting traveling salesman and related problems. In *Handbook of Approximation Algorithms and Metaheuristics*, pages 611–628. Chapman and Hall/CRC, 2018. doi:10.1201/9781351236423-34.

3 Julian Blank, Kalyanmoy Deb, and Sanaz Mostaghim. Solving the bi-objective traveling thief problem with multi-objective evolutionary algorithms. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 46–60. Springer, 2017. doi:10.1007/978-3-319-54157-0_4.

4 Adrian Bock and Laura Sanità. The capacitated orienteering problem. *Discrete Applied Mathematics*, 195:31–42, 2015. doi:10.1016/J.DAM.2014.10.001.

5 M. R. Bonyadi, Z. Michalewicz, and L. Barone. The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. In *2013 IEEE Congress on Evolutionary Computation*, pages 1037–1044, 2013. doi:10.1109/CEC.2013.6557681.

6 Jakob Bossek, Katrin Casel, Pascal Kerschke, and Frank Neumann. The node weight dependent traveling salesperson problem: approximation algorithms and randomized search heuristics. In *Genetic and Evolutionary Computation Conference, GECCO 2020*, pages 1286–1294. ACM, 2020. doi:10.1145/3377930.3390243.

7 Jonatas B.C. Chagas and Markus Wagner. A weighted-sum method for solving the bi-objective traveling thief problem. *Computers & Operations Research*, 138:105560, 2022. doi:10.1016/j.cor.2021.105560.

8 Kamalika Chaudhuri, Brighten Godfrey, Satish Rao, and Kunal Talwar. Paths, trees, and minimum latency tours. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 36–45. IEEE, 2003. doi:10.1109/SFCS.2003.1238179.

9 Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms (TALG)*, 8(3), 2012. doi:10.1145/2229163.2229167.

10 Leah Epstein, Asaf Levin, Alberto Marchetti-Spaccamela, Nicole Megow, Julián Mestre, Martin Skutella, and Leen Stougie. Universal sequencing on a single machine. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 230–243. Springer, 2010. doi:10.1007/978-3-642-13036-6_18.

11 Jan Eube, Kelin Luo, Aneta Neumann, Frank Neumann, and Heiko Röglin. Effective traveling for metric instances of the traveling thief problem, 2026. doi:10.48550/arXiv.2604.19271.

12 Jan Eube, Kelin Luo, Heiko Röglin, and Sarah Sturm. Approximation algorithms for the traveling thief problem, 2026. arXiv:2607.05164.

13 Zuguang Gao, John R. Birge, and Varun Gupta. Approximation schemes for multiperiod binary knapsack problems. In *Computer Science - Theory and Applications - 16th International Computer Science Symposium in Russia, CSR 2021, Sochi, Russia, June 28 - July 2, 2021, Proceedings*, 2021. doi:10.1007/978-3-030-79416-3_8.

14 Daniel Herring, Michael Kirley, and Xin Yao. A comparative study of evolutionary approaches to the bi-objective dynamic travelling thief problem. *Swarm Evol. Comput.*, 84:101433, 2024. doi:10.1016/J.SWEVO.2023.101433.

15 Frank Neumann, Sergey Polyakovskiy, Martin Skutella, Leen Stougie, and Junhua Wu. A fully polynomial time approximation scheme for packing while traveling. In *ALGOCLOUD*, volume 11409 of *Lecture Notes in Computer Science*, pages 59–72. Springer, 2018. doi:10.1007/978-3-030-19759-9_5.

16 Adel Nikfarjam, Aneta Neumann, and Frank Neumann. On the use of quality diversity algorithms for the travelling thief problem. *ACM Trans. Evol. Learn. Optim.*, 4(2):12, 2024. doi:10.1145/3641109.

17 Sergey Polyakovskiy, Mohammad Reza Bonyadi, Markus Wagner, Zbigniew Michalewicz, and Frank Neumann. A comprehensive benchmark set and heuristics for the traveling thief problem. In *Genetic and Evolutionary Computation Conference, GECCO 2014*, pages 477–484. ACM, 2014. doi:10.1145/2576768.2598249.

18 Michal R. Przybylek, Adam Wierzbicki, and Zbigniew Michalewicz. Decomposition algorithms for a multi-hard problem. *Evol. Comput.*, 26(3), 2018. doi:10.1162/EVCO_A_00211.

19 Junhua Wu, Sergey Polyakovskiy, Markus Wagner, and Frank Neumann. Evolutionary computation plus dynamic programming for the bi-objective travelling thief problem. In *Proceedings of the genetic and evolutionary computation conference*, pages 777–784, 2018. doi:10.1145/3205455.3205488.

20 Wenzheng Xu, Zichuan Xu, Jian Peng, Weifa Liang, Tang Liu, Xiaohua Jia, and {Sajal K.} Das. Approximation algorithms for the team orienteering problem. *Proceedings - IEEE INFOCOM*, 2020. doi:10.1109/INFOCOM41043.2020.9155343.

21 Mohamed El Yafrani, Shelvin Chand, Aneta Neumann, Belaïd Ahiod, and Markus Wagner. Multi-objectiveness in the single-objective traveling thief problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 107–108, 2017. doi:10.1145/3067695.3076010.

A A $(2e + \epsilon)$-approximation Algorithm for Weighted TSP

In this section we describe how the approximation factor for WTSP can be improved to $(2e + \epsilon)$. First, we provide a randomized algorithm with this guarantee (following the same ideas as the randomized algorithm for universal sequencing [10]). Then we describe how this algorithm can be derandomized. First, we generalize Lemma 4 for a randomized setting:

▶ **Lemma 26.** *Let a randomized algorithm be given that produces a TSP tour π and let π^* be the optimum WTSP tour. Let c be an arbitrary constant. If it holds for any weight W that $E\left[D^{\pi}(W)\right] \leq c \cdot D^{\pi^*}(W)$ then $E\left[cost(\pi)\right] \leq c \cdot cost(\pi^*)$.*

Proof. Let us for simplicity define $f(-1) = 0$. For any tour π we can express $\text{cost}(\pi)$ as follows:

$$\text{cost}(\pi) = \sum_{W=0}^{w(V)} D^{\pi^*}(W) \cdot (f(W) - f(W-1))$$

Using the linearity of expectation we obtain:

$$\begin{aligned} E[\text{cost}(\pi)] &= E\left[\sum_{W=0}^{w(V)} D^{\pi}(W) \cdot (f(W) - f(W-1))\right] \\ &= \sum_{W=0}^{w(V)} E[D^{\pi}(W)] \cdot (f(W) - f(W-1)) \\ &\leq \sum_{W=0}^{w(V)} c \cdot D^{\pi^*}(W) \cdot (f(W) - f(W-1)) \\ &= c \cdot \text{cost}(\pi^*) \end{aligned}$$

◀

Using this, we can improve the approximation factor of Algorithm 1 by sampling a number λ uniformly from the interval $[0, 1)$ and modify the choice of w_s of the algorithm such that:

$$\ell(\text{QTSP}(r, (R, d), w_s)) \leq e^{s+\lambda} < \ell(\text{QTSP}(r, (R, d), w_s + 1))$$

Similarly to Lemma 7, one can show that:

▶ **Lemma 27.** *If* $(2+\epsilon)\cdot D^{\pi^*}(\mathcal{W}) \leq e^{j+\lambda}$ *then* $W_j^{\psi} \geq w(v) - \mathcal{W} + 1$.

Under this circumstances the value $D^{\pi^*}(\mathcal{W})$ can be bounded by the length of the tours $\psi_1, \ldots, \psi_j$. By summing up the respective lengths and applying the harmonic sum we obtain:

▶ **Lemma 28.** *If* $(2+\epsilon)\cdot D^{\pi^*}(\mathcal{W}) \leq e^{j+\lambda}$ *then* $D^{\pi}(\mathcal{W}) \leq e^{j+\lambda}\cdot\frac{e}{e-1}$

This lemma can then be used to bound the expected approximation factor of the randomized algorithm.

▶ **Theorem 29.** *It holds that* $E[cost(\pi)] \leq (2+\epsilon)\cdot e \cdot cost(\pi^*)$.

Proof. Let $\mathcal{W} \in w(V)$ be an arbitrary weight and let $c = \log_e((2+\epsilon)\cdot D^{\pi^*}(\mathcal{W}))$. We can then use Lemma 28 to bound the expected value of $D^{\pi}(\mathcal{W})$ as follows:

$$\begin{aligned}
E_{\lambda\sim U[0,1]}[D^{\pi}(\mathcal{W})] &\leq E_{\lambda\sim U[0,1]}\left[\min_{j\in\mathbb{N}_0: j+\lambda\geq c} e^{j+\lambda}\cdot\frac{e}{e-1}\right] \\
&= e^c\cdot\frac{e}{e-1}E_{\lambda\sim U[0,1]}\left[\min_{j\in\mathbb{N}_0: j+\lambda\geq c} e^{j+\lambda}/e^c\right] \\
&= (2+\epsilon)\cdot D^{\pi^*}(\mathcal{W})\cdot\frac{e}{e-1}\int_0^1 e^{\lambda'}d\lambda' \\
&= (2+\epsilon)\cdot e\cdot D^{\pi^*}(\mathcal{W})
\end{aligned}$$

Since this holds for every arbitrary weight $\mathcal{W} \in w(V)$ we can directly apply Lemma 26 and the theorem is proven. ◀

We can derandomize the algorithm by choosing an integer value $l \in \mathbb{N}$ and execute the randomized algorithm with every value $\lambda \in \left\{\frac{0}{l}, \frac{1}{l}, \ldots, \frac{l-1}{l}\right\}$ and taking the solution with the smallest cost. It is easy to verify that if we had chosen the value λ from $\left\{\frac{0}{l}, \frac{1}{l}, \ldots, \frac{l-1}{l}\right\}$ uniformly at random the expected cost of the solution can be bounded by $(1+\frac{1}{l})\cdot(2+\epsilon)\cdot e\cdot \text{cost}(\pi^*)$. Thus, for at least one of the possible choices the cost of the tour must lie below this expected value. By using a smaller value $\epsilon' < \epsilon$ instead of ϵ in this algorithm and choosing a sufficiently large l one obtains an $(2e+\epsilon)$-approximation guarantee:

▶ **Theorem 1.** *For any* $\epsilon > 0$*, one can calculate a* $(2e+\epsilon)$*-approximation for* Wtsp *in polynomial time if all weights are polynomially bounded integers.*

B Applying our Approach to Different Models

We introduced the Cttp model in this paper because it simultaneously captures the general challenges of the traveling thief problem while it also simplifies writing a lot compared to existing models. In this section we explain how our Cttp algorithm can be adapted to also approximate different settings. In particular, we argue that our algorithm can be used to calculate an approximate Pareto set for Bttp and that this is also true if the starting node is not fixed, if there can be multiple items at the same location or if the agent is required to visit every location even if it does not want to collect any items there.

B.1 Bi-objective TTP

In Bttp we are not given a desired profit $\mathcal{P}$ and instead one aims to find a Pareto set with respect to the profit $p(\pi)$ and the travel time $\text{cost}(\pi)$ of the tours. A tour π is Pareto-optimal if there exists no tour π' such that $p(\pi') \geq p(\pi)$ and $\text{cost}(\pi') \leq \text{cost}(\pi)$, with at least one

of these inequalities being strict. The Pareto set is then the set of all Pareto-optimal tours. Since both objectives are NP-hard, computing this set is also NP-hard. We therefore restrict our attention to the computation of an approximate Pareto set. For two values $\alpha_1, \alpha_2 \in \mathbb{R}_{>0}$, we say that a set S of tours is an (α_1, α_2)-approximate Pareto set if for every feasible tour π there exists a tour $\pi' \in S$ such that $\alpha_1 \cdot p(\pi') \geq p(\pi)$ and $\text{cost}(\pi') \leq \alpha_2 \cdot \text{cost}(\pi)$.

If we are given a bicriteria (α_1, α_2)-approximation algorithm for CTTP and an $\epsilon > 0$, we can obtain a $((1+\epsilon)\cdot\alpha_1, \alpha_2)$-approximate Pareto set for BTTP by simply applying the bicriteria approximation algorithm with $\mathcal{P} = (1+\epsilon)^i$ for every $i \in \mathbb{N}_0$ with $i \leq \lceil \log_{1+\epsilon}(p(V)) \rceil$, where $p(V) = \sum_{v \in V} p(v)$. One may verify that for a constant $\epsilon > 0$ the choices of i are polynomial in the input size. Thus, we can use our CTTP algorithm to calculate a $(9+\epsilon, 9+\epsilon)$-approximate Pareto set for BTTP.

B.2 Visiting every Location

In most of the existing literature the agent is required to visit every single location, even if it does not collect an item there. One can deal with this variant by using Christofides 1.5–approximation algorithm for TSP [8] to calculate a tour visiting all locations where the agent does not collect any items and adding this to the start of the CTTP tour produced by our algorithm. Since the agent still travels with its maximum speed while visiting these locations, the travel time of this initial subtour is bounded by 1.5 times the travel time of any solution that visits every node and the approximation factor α_2 gets increased by at most 1.5.

B.3 Unknown Starting Location

If the initial position r of the agent is not given, we can simply evaluate all n choices of r and choose the best one. Since our algorithm requires that there is no item placed at r we have to take care of the respective item in advance. We can simply run the algorithm for both the case that the item gets picked up and that we do not pick up the item and take the better solution. There are two models in the literature regarding the point in time at which the item at the starting position needs to be collected (if we decide to collect it). If the agent is required to pick up the item in the beginning, we can simply modify f and $w_{\max}$ to incorporate that the agent already starts with an initial weight. If the item can be picked up in the end, we only have to modify $w_{\max}$ to make sure that the agent can still collect the item when finishing the tour.

Doing this we are able to obtain an approximate CTTP solution. If we want to calculate an approximate Pareto set for BTTP we need to calculate the respective Pareto sets for every possible starting node and then merge them to obtain an approximate Pareto set for the situation that r is not fixed.

B.4 Having Multiple Items at the Same Location

If multiple items are placed at the same location we can simply create multiple copies of the respective node and place exactly one item at each of these nodes. It could happen that the solution calculated by our algorithm visits different copies of a location at different points in time. In the classic TTP model one is required to visit every location at most once and needs to decide then which items are collected. We can handle this situation by modifying our tour such that we only visit a location when we visit the last copy of it. Once we visit it we collect all items that are placed at the copies visited by our original tour. This only causes some of the weights to be collected later which cannot increase the travel time.

However, we assume that the minimum distance between r and any other location is at least 1. If there are multiple items placed at the starting location, we would need to have copies of r that are at distance 0 to it. If we are allowed to collect the items at the starting location at the end of the tour this does not cause too much problems. We simply have to modify Algorithm 2 to also consider $\psi' = \textsc{Cop}((V \setminus P_i, d), 0, T_{i+1} - T_i)$ as a possible choice for subtour ψ_i for every $i \in \{0, \dots, s\}$. We also have to be a little bit more careful during some parts of our analysis, but this is mainly due to the fact that we want to avoid to divide by 0. In principle the algorithm works as intended.

If we are required to collect the items placed at r in the beginning of the tour, things get a little bit more complicated. For a given set S' of the items that get collected at r, we can simply modify the function f and the value of $w_{\max}$ to incorporate that the agent already starts with some weight. To do this, we need to be able to predetermine S'. We describe in the full version of this paper how this can be achieved.

C Omitted Proofs

C.1 Proof of Lemma 14

▶ **Lemma 14.** *If $p(B) \leq \frac{2+\epsilon}{9+3\epsilon}\mathcal{P}$ it holds for any $i \in [s] \cup \{0\}$ that if $p(P_i) < \frac{\mathcal{P}}{9+3\epsilon}$ then there exists an $i' \geq i$ such that the interval $[T_{i'} + 1, T_{i'+1}]$ is efficient after P_i has been collected.*

Proof. For simplicity we say that an interval is efficient if it is efficient after P_i has been collected. We subdivide the intervals into three categories:

- $I_{inef} = \{j \in [s] \cup \{0\} \mid [T_j + 1, T_{j+1}]$ is not efficient.$\}$ contains all the indices corresponding to inefficient intervals.
- $I_{block} := \{j \in [i-1] \cup \{0\} \mid [T_j + 1, T_{j+1}]$ is efficient.$\}$ contains all indices smaller i corresponding to efficient intervals. We call the respective intervals blocked.
- $I_{eff} := \{j \in [s] \cup \{0\} \mid j \geq i \wedge [T_j + 1, T_{j+1}]$ is efficient.$\}$ contains all indices with value at least i corresponding to efficient intervals.

Let $p(P_i) < \frac{\mathcal{P}}{9+3\epsilon}$. Since every item on the optimum tour is contained in B or in exactly one of the intervals we know that:

$$\sum_{j \in I_{inef}} p(A_j \setminus P_i) + \sum_{j \in I_{block}} p(A_j \setminus P_i) + \sum_{j \in I_{eff}} p(A_j \setminus P_i)$$
$$= \sum_{j=0}^{s} p(A_j \setminus P_i) \geq \left(\sum_{j=0}^{s} p(A_j) \right) - p(P_i) > \frac{7+2\epsilon}{9+3\epsilon}\mathcal{P} - \frac{\mathcal{P}}{9+3\epsilon} = \frac{2}{3}\mathcal{P}$$

We want to bound the contribution of the inefficient and the blocked intervals to the left side of the inequality. For any $j \in I_{inef}$ it holds that:

$$p(A_j \setminus P_i) < \frac{\mathcal{P}}{12\mathcal{D}} \cdot D^{\pi^*}(T_j + 1)2^{j+1} = \frac{\mathcal{P}}{3\mathcal{D}} D^{\pi^*}(T_j + 1)2^{j-1}$$

By combining this with Lemma 10 we obtain:

$$\begin{aligned}\sum_{j\in I_{inef}} p(A_j \setminus P_i) &< \frac{\mathcal{P}}{3\mathcal{D}} \sum_{j\in I_{inef}} D^{\pi^*}(T_j+1)2^{j-1} \\ &\leq \frac{\mathcal{P}}{3\mathcal{D}} \sum_{j=0}^{s} D^{\pi^*}(T_j+1)2^{j-1} \\ &\overset{\text{Lem. 10}}{\leq} \frac{\mathcal{P}}{3\mathcal{D}}\text{cost}(\pi^*) \\ &\leq \frac{\mathcal{P}}{3}\end{aligned}$$

Now we consider the blocked intervals. Let $j \in I_{block}$. Since $j \leq i-1$ it holds that $P_j \subseteq P_i$. Thus, the interval $[T_j+1, T_{j+1}]$ was efficient after collecting P_j and by Lemma 13:

$$p(\psi_j) \geq \frac{p(A_j \setminus P_j)}{3+\epsilon} \geq \frac{p(A_j \setminus P_i)}{3+\epsilon}.$$

By summing up over all blocked intervals we obtain:

$$\begin{aligned}\sum_{j\in I_{block}} p(A_j \setminus P_i) &\leq (3+\epsilon) \sum_{j\in I_{block}} p(\psi_j) \\ &\leq (3+\epsilon) \sum_{j=0}^{i-1} p(\psi_j) \\ &= (3+\epsilon) \cdot p(P_i) \\ &< (3+\epsilon) \cdot \frac{\mathcal{P}}{9+3\epsilon} \\ &= \frac{\mathcal{P}}{3}\end{aligned}$$

By combining all of these bounds we may conclude that I_{eff} cannot be empty:

$$\sum_{j\in I_{eff}} p(A_j \setminus P_i) > \frac{2}{3}\mathcal{P} - \sum_{j\in I_{inef}} p(A_j \setminus P_i) - \sum_{j\in I_{block}} p(A_j \setminus P_i) > \frac{2}{3}\mathcal{P} - \frac{\mathcal{P}}{3} - \frac{\mathcal{P}}{3} = 0.$$

This directly proves the lemma. ◀

C.2 Proof of Lemma 23

▶ **Lemma 23.** *If $p(B) \geq \frac{2+\epsilon}{9+3\epsilon}\mathcal{P}$ then it holds that $p(\pi^{(2)}) \geq \frac{\mathcal{P}}{9+3\epsilon}$.*

Proof. By Lemma 22, we know that $p(\sigma) \geq \frac{\mathcal{P}}{9+3\epsilon}$. Thus, it can only happen that $p(\pi^{(2)}) < \frac{\mathcal{P}}{9+3\epsilon}$ if there exists an $l' \in [l-1]$ such that $\pi^{(2)} = (r, \sigma_1, \ldots, \sigma_{l'})$ and $\text{cost}(\pi^{(2)}) > 6\mathcal{D}$. We may then lower-bound $p(\pi^{(2)})$ as follows:

$$\begin{aligned}
p(\pi^{(2)}) &= \sum_{j=1}^{l'} p(\sigma_j) \\
&\geq \sum_{j=1}^{l'} \frac{1}{t_2} \cdot \mathrm{cost}((r, \sigma_j, r), s(\sigma_j)) \\
&\geq \frac{1}{t_2} \sum_{j=1}^{l'} \mathrm{cost}\left((r, \sigma_j, r), \sum_{j'=1}^{j-1} w(\sigma_{j'})\right) \\
&\geq \frac{1}{t_2} \cdot \mathrm{cost}(\pi^{(2)}) \\
&> \frac{\mathcal{P}}{6 \cdot (9 + 3\epsilon) \cdot \mathcal{D}} \cdot 6\mathcal{D} \\
&= \frac{\mathcal{P}}{9 + 3\epsilon}.
\end{aligned}$$

This proves the lemma. ◀