Polynomial Time Approximation Schemes for the Traveling Repairman and Other Minimum Latency Problems

Sitters, Rene

published in
SIAM Journal on Computing
2021

DOI (link to publisher)
10.1137/19M126918X

document version
Publisher's PDF, also known as Version of record

SIAM J. COMPUT.
Vol. 50, No. 5, pp. 1580–1602

POLYNOMIAL TIME APPROXIMATION SCHEMES FOR THE TRAVELING REPAIRMAN AND OTHER MINIMUM LATENCY PROBLEMS*

RENÉ SITTERS†

Abstract. We give a polynomial time approximation scheme for the weighted traveling repairman problem (TRP) in the Euclidean plane, on trees, and on planar graphs. This improves upon the quasi-polynomial time approximation schemes for the unweighted TRP in the Euclidean plane and trees and on the 3.59-approximation for planar graphs. The algorithms are based on a new decomposition technique that reduces the approximation of weighted TRP to instances for which we may restrict ourselves to solutions that are the concatenation of only a constant number of traveling salesman problem paths. A similar reduction applies to many other problems with an average completion time objective. To illustrate the strength of this approach, we apply the same technique to the well-studied scheduling problem of minimizing total weighted completion time under precedence constraints, $1|prec|\sum w_jC_j$, and present a polynomial time approximation scheme for the case of interval order precedence constraints. This improves on the known 3/2-approximation for this problem.

Key words. minimum latency tour, traveling repairman, approximation algorithms, interval orders, total completion time

AMS subject classifications. 68W25, 90C27, 90C59, 90B06, 05C38

DOI. 10.1137/19M126918X

1. Introduction.

The traveling repairman problem (TRP) (also known as the minimum latency problem) is similar to the well-known traveling salesman problem (TSP). An instance is given by a set of points $V \cup \{r\}$ in a metric space, and a feasible solution is a path Π, starting at origin r, that visits each of the points in V. The *completion time* $C(v)$ of a point v is the distance from r to v on path Π, and the objective is to minimize the total completion time of the points. Hence, the problem can be seen as a traveling repairman whose aim is to minimize the average time of arrival at the clients. The traveling salesman, on the other hand, aims at minimizing the travel time of the salesman itself.

The approximability of the TRP has been the subject of many papers [1, 3, 4, 11, 13, 14, 15, 17, 19, 20, 25, 28, 29, 34]. The first approximation ratio, given by Blum et al. [13], was 144, and the current smallest ratio for general metrics is 3.59 by Chaudhuri et al. [14]. Even for edge-weighted trees, the best polynomial time approximation ratio was 3.59 [20] until Archer and Blasiak reduced it to 3.03 [3]. $\mathcal{NP}$-hardness for the tree case was proven in the conference paper [29]. For the Euclidean plane, a 3.59-approximation follows from the TRP algorithm by Goemans and Kleinberg [20] in combination with the polynomial time approximation scheme (PTAS) for the Euclidean k-TSP by Arora [6]. A quasi-polynomial time approximation scheme (QPTAS) for the TRP on edge-weighted trees and the Euclidean plane

*Received by the editors July 2, 2019; accepted for publication (in revised form) July 19, 2021; published electronically October 21, 2021. A preliminary version of this paper appeared in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2014, pp. 604–616.

https://doi.org/10.1137/19M126918X

†Department of Econometrics and Operations Research, Vrije Universiteit, Amsterdam, 1081 HV, The Netherlands, and Centrum Wiskunde & Informatica (CWI), Amsterdam, 1090 GB, The Netherlands (r.a.sitters@vu.nl).

was given by Arora and Karakostas [9, 11]. Arora and Karakostas write that they "do not know whether the running time can be reduced to polynomial." Here we show that this is indeed possible. The authors remark that their QPTAS also applies to edge-weighted planar graphs. However, this claim turned out to be incorrect and it is not known how to fix the issue in the original proof [22]. The general decomposition technique that we introduce here not only reduces the running time to polynomial but also does not suffer from the same issue. Hence, we do get a PTAS for edge-weighted planar graphs.

In the *weighted* TRP, each point v has a weight w_v and the goal is to minimize the sum of weighted completion times: $\sum_v w_v C(v)$. In our terminology, TRP is unweighted by default. On the other hand, we assume by default that for graph metrics, each edge has a given *length* and the *distance* between any two points is given by the length of the shortest path between the two points. Hence, we refer to the TRP on edge-weighted trees simply as the TRP on trees and call it weighted TRP on trees if, additionally, each vertex v has a weight w_v. In this paper, we study the weighted TRP.

Many approximation algorithms for the TRP solve some form of the k-TSP or k-MST as a subroutine. In the k-TSP (k-MST), one needs to find the shortest tour (tree) that visits at least k of the n input points. For example, the algorithm by Goemans and Kleinberg [20] first computes approximate k-TSP tours for all $k \leqslant n$ and then combines a subset of these tours into one TRP solution. A similar approach is to make subtours of geometrically increasing length and to visit a maximum number of points in each subtour. The obtained ratio is 3.59α [20], where α is the approximation ratio of the k-TSP (or k-MST). So far, the best ratio for k-TSP and k-MST is $2 + \epsilon$ [10]. Chaudhuri et al. [14] found a way to bypass this factor 2 in the analysis and noted that breaking the barrier of 3.59 would probably involve an approach different from combining small tours. Indeed, the exact algorithm for the TRP on the line [1] and the QPTAS [11] for the plane and trees are different since they find a solution directly by one dynamic program.

In this paper, we introduce a decomposition technique which reduces the TRP to what we call a δ-*bounded* TRP. (See Algorithm DECOMPOSE in section 2.1 and Theorem 2.11). We say that a TRP instance I is δ-bounded if it has the additional restriction that the tour is not allowed to start before some time $L > 0$ and it has the property that the length of the shortest tour on all points is at most δL. The strength of the reduction is that with loss of a factor $(1 + \epsilon)$ in the approximation we may assume that δ is a number that depends on ϵ only. We shall say that a set of instances is *bounded* if all instances in the set are δ-bounded for some constant δ. For δ-bounded instances, we can restrict ourselves to solutions that are a concatenation of κ TSP paths where $\kappa = O(\log \delta / \log \epsilon)$, which is in our case a constant depending on ϵ only. Consequently, for bounded instances, the quasi-polynomial time algorithms from [9, 11] for the TRP in the Euclidean plane and on edge-weighted trees can be easily turned into polynomial time algorithms. Hence, we obtain a PTAS for the TRP on these metrics. For planar graphs, the PTAS now follows from the planar-TSP PTAS [8] and a more recent result by Klein [23] on subset spanners for planar graphs. For problems where the bounded instances are just as hard to approximate as the general problem, as in the min sum set cover problem [18], for example, the decomposition does not give an improved ratio.

As mentioned, in the weighted TRP each point v has a weight w_v and the goal is to minimize the total weighted completion time $\sum_v w_v C(v)$. If weights are integer, then one might (depending on the metric space) replace a point of weight w_v by w_v points

at distance zero. However, this reduction is not polynomial in general since weights can be exponentially large in the input size. Rounding the weights is not an option since, for example, a point of weight 2^n visited at time 1 contributes the same to the objective as a point of weight 1 visited at time 2^n. Some approximation algorithms also work for arbitrary weights (see, e.g. [20] or [4]), while other results are reported to hold only for the unweighted (or polynomially bounded weights) case. In the QPTASs from [11], the running time for weighted instances is $(nW)^{O(\log n \log W)/\epsilon}$ for trees and $(nW)^{O(\log W)/\epsilon}$ for the Euclidean plane, where W is the maximum weight of a point. We show that for any metric space and constant $\epsilon > 0$, *any* α-approximation algorithm for the TRP with polynomially bounded weights yields (with only a polynomial factor increase in the running time) an $\alpha(1+\epsilon)$-approximation for weighted TRP in the same metric space. (See Theorem 2.12.)

In the conference version [30] of this paper, the approximation schemes were of polynomial time under the assumption that vertex weights are polynomially bounded. In [30], there was no explicit decomposition of the input points, but instead, all subproblems were defined on the complete set of points and the final solution was obtained by an additional dynamic program over all subproblems. Further, the claimed result for planar graphs turned out to be invalid since it had the same flaw as [11]. (See the note in [31].) The decomposition into problems on subsets presented here, together with the use of subset spanners for planar graphs by Klein [23], fixes the flaw.

The decomposition approach can be applied to many sequencing problems with a minimum total (weighted) completion time objective. This is illustrated by our second application, which is the scheduling problem of minimizing total weighted completion time under precedence constraints, known as $1|prec|\sum w_jC_j$ in the standard scheduling notation. (See, e.g., Graham et al. [21].) The approximability of this problem has been studied in many papers. The problem is known to be $\mathcal{NP}$-hard [26, 27], and several 2-approximation algorithms are known. Bansal and Khot [12] showed that no $(2-\epsilon)$-approximation algorithm exists under the assumption that some variant of the unique games conjecture is true. We present a PTAS for the case of interval order precedence constraints. (See Algorithm DECOMPOSE in section 3.) Woeginger [35] gave a 1.62-approximation algorithm, and a 3/2-approximation was given by Ambühl et al. [2]. Somewhat surprisingly, the same paper shows that scheduling interval orders is in fact $\mathcal{NP}$-hard. Hence, our PTAS closes the gap in the approximability for this problem.

Recently, the same technique has been applied to give a $(2+\epsilon)$-approximation for the single machine scheduling problem with release times and arbitrary precedence constraints, $1|r_j, prec|\sum w_jC_j$ [32], improving on the 2.542-approximation algorithm from [33].

α-approximation schemes. We say that an algorithm A for a minimization problem is an α-approximation algorithm if for any instance of the problem the value (or cost) of the computed solution is no more than α times the optimal value. We say that a family of algorithms A_ϵ is an α-approximation scheme if for every constant $\epsilon > 0$, algorithm A_ϵ is a $(1+\epsilon)\alpha$-approximation algorithm. The scheme is said to run in polynomial time if A_ϵ runs in polynomial time for all $\epsilon > 0$. When $\alpha = 1$, this is known as a PTAS.

Outline. In section 2.1, we describe and analyze the decomposition technique for the TRP. In section 2.2, we show how to reduce the subproblems to their unweighted versions. In sections 2.3 to 2.5, we give approximation schemes for the subproblem on trees, the plane, and planar graphs. In section 3.1, we apply the same decomposition technique to the single machine scheduling problem, and in section 3.2 we give the

PTAS for the reduced problem.

2. The traveling repairman problem.

DEFINITION 2.1. *An instance of the weighted traveling repairman problem (TRP) is given by set $V \cup \{r\}$ of $n+1$ points, where point r is referred to as the origin, and a symmetric distance matrix $[d_{v,w}]$. Distances are nonnegative integers and satisfy the triangle inequality. There is an integer weight $w_v \geq 0$ for each point $v \in V$. For the unweighted problem, $w_v = 1$ for all $v \in V$. A solution is given by a permutation $\Pi = (v_1, \ldots, v_n)$ of V. The completion time $C(v)$ of a point $v = v_j$ is the distance from r to v on the path defined by Π: $C(v_j) = d_{r,v_1} + \sum_{i=2}^{j} d_{v_{i-1},v_i}$. The value of a solution is its total weighted completion time, $\sum_{v \in V} w_v C(v)$, and the goal is to find a solution of minimum value.*

For the weighted TRP, we may assume that $V \cup \{r\}$ with distance d is a metric space since points at distance zero can be merged by adding their weights. For the unweighted version, the definition is more subtle. When two points are at zero distance, we formally no longer have a metric space. Typically, the complexity of the unweighted TRP is polynomially equivalent to that of the weighted TRP with polynomially bounded weights if any point of weight w can be replaced by w points at a small enough distance. For discrete metric spaces, like a uniform metric space, this reduction may not be possible. Note, for example, that the tree TRP with uniform edge weights is easy but is NP-hard for general, polynomially bounded, edges weights [29].

It is convenient for the analysis to see a solution Π as a tour that starts at time zero at the origin, ends at the origin, and is traversed with unit speed. That means we assume that there is a continuous path of length d_{v_{j-1},v_j} between consecutive points v_{j-1} and v_j in Π. Any walk that starts and ends in the origin and visits each point at least once is considered a solution. The completion time $C(v)$ of input point v is then defined as the time at which v is visited for the first time. The completion time of the tour is the first moment at which the tour is back in the origin and all points are visited. Further, we assume that $d_{r,v} \geq 1$ for all $v \in V$ and, hence, $C(v) \geq 1$. We assume $0 < \epsilon \leq 1$ and use the notation $O_\epsilon(.)$ when ϵ is assumed constant. For example, $n/\epsilon = O_\epsilon(n)$ and $n^{1/\epsilon^2} = n^{O_\epsilon(1)}$.

2.1. A decomposition theorem.

We will show how to partition the given point set such that the TRP instances induced by the partition are in a way easier to approximate than the original instance and such that solutions can be computed independently and then concatenated in a fixed order with loss of just a factor $(1+\epsilon)$ in the approximation ratio. All it takes to make the partition is any constant factor β-approximation algorithm for the TRP (such as [14] or [13]). Although the partition and running time depend on the ratio β, the loss in the ratio does not. Any constant factor β-approximation algorithm for the TRP together with an α-approximation for each of the reduced instances implies a $(1+\epsilon)\alpha$-approximation for the original instances.

The decomposition and concatenation are described by Algorithm DECOMPOSE below. For a proper definition of the algorithm, we first give the following general definition and lemma. Then, given the formulation of the algorithm, it follows from Lemma 2.4 that the algorithm is indeed well-defined.

DEFINITION 2.2. *For a given number $L \geq 0$, we say that a TRP instance (as defined in Definition* 2.1*) is* L-delayed *if it has the additional restriction that the solution should start in the origin at time L instead of time* 0.

LEMMA 2.3. *Let I be an L-delayed instance, and let σ' be a solution (a tour) for I of value Z. Further, let σ^* be a path, starting at the origin, of length at most δL that visits all points. Then, given σ^* and σ', we can find, for any $\epsilon > 0$, a solution for I of value at most $(1+\epsilon)Z$ that is back in the origin by time $(1+5\delta/\epsilon)L$.*

Proof. Follow solution σ' until time $t = (1+2\delta/\epsilon)L$. Then return to the origin and follow path σ^* and return to the origin after all points are visited. Since no point is more than δL away from the origin, the completion time of this solution is no more than $t + 3\delta L = (1 + 2\delta/\epsilon + 3\delta)L \leq L + 5\delta L/\epsilon$ (using $\epsilon \leq 1$).

Comparing the completion times in the new solution with that of σ', we see that the completion time of any point visited before t remains the same. Any point visited after t by σ' is visited before time $t + 2\delta L$ by the new solution. Hence, for any point the increase in completion time is no more than a factor $(t + 2\delta L)/t = 1 + 2\delta L/t \leq 1 + (2\delta + \epsilon)L/t = 1 + \epsilon$. ∎

In step 1 of Algorithm DECOMPOSE, we run any constant factor β-approximation algorithm and label the vertices by their completion times. The completion times are then used to make a random partition $V_1, V_2, \ldots$ of the point set V in step 2. For each subset V_i, the induced TRP is approximated independently in step 3, and the subtours are concatenated in step 4.

The decomposition principle is simple and applies to many other minimum latency problems. Note that the partition in step 2 depends on β and ϵ but also on a parameter γ. Intuitively, γ gives an upper bound on the ratio between the length of a near optimal solution for an instance of the subproblem and the length of a minimum-length solution for that instance. The choice of γ is problem-specific, and for the TRP we can bound this by $\gamma = 10/\epsilon$. (See Lemma 2.4.)

Algorithm DECOMPOSE (TSP):

1) **Approximate:** Apply any constant factor β-approximation algorithm for the problem. Let $1 \leq C_1 \leq \cdots \leq C_n$ be the completion times in the solution, and label the points in V by $1, 2, \ldots, n$ such that C_j is the completion time of point j for all $j \in \{1, 2, \ldots, n\}$.
2) **Partition:** Let $a = \beta\gamma/\epsilon$, and take b uniformly at random in $[0, a]$, where $\gamma = 10/\epsilon$.
   Let $t_i = e^{a(i-2)+b}$ for $i = 1, 2, \ldots, q+1$, where q is large enough such that $C_n < t_{q+1}$. Partition the points into sets $V_i = \{j | t_i \leq C_j < t_{i+1}\}$, $i \in \{1, 2, \ldots, q\}$.

3a) **Approximate subproblems:** Let I_i be the L_i-delayed instance restricted to the points in $\{r\} \cup V_i$ and with a delay of $L_i = \gamma t_i$. For each I_i, find an approximate schedule σ_i.

3b) **Modify:** For each i, modify σ_i such that it completes before time γt_{i+1} and its value is increased by at most a factor $(1+\epsilon)$. (See Lemma 2.4.)

4) **Concatenate:** Return the concatenation of $\sigma_1, \ldots, \sigma_q$ as the final solution.

Given any β-approximation algorithm, step 1 is well-defined. Note that the partition in step 2 is proper since $t_1 = e^{b-a} \leq e^0 = 1 \leq C_1$. The next lemma shows that the modification in step 3b can be done with only a factor $(1+\epsilon)$ loss in the objective. After the modification, any tour σ_i starts at time γt_i and completes before time γt_{i+1}. Hence, the union of these solutions is feasible for instance I. Additionally, we may remove the idle time by letting tour σ_i start at the moment that σ_{i-1} completes.

LEMMA 2.4. *Given any α'-approximate solution for I_i, we are able to find an $\alpha = (1+\epsilon)\alpha'$-approximate solution for I_i that returns to the origin before time γt_{i+1}.*

Proof. In step 1, we computed a path of length at most $t_{i+1} = e^a t_i$ that visits all points of V_i. Now apply Lemma 2.3 with $L_i = \gamma t_i$ and $\delta = t_{i+1}/L_i = e^a/\gamma = e^a\epsilon/10$. The lemma states that we can find a $(1+\epsilon)\alpha'$-approximate solution for I_i that returns to the origin before time

$$(1 + 5\delta/\epsilon)L_i = (1 + e^a/2)L_i = (1 + e^a/2)\gamma t_i < e^a \gamma t_i = \gamma t_{i+1},$$

where the inequality follows from $1 < e^a/2$. Indeed, $a = \beta\gamma/\epsilon = 10\beta/\epsilon^2 \geq 10$, implying $1 < e^{10}/2 \leq e^a/2$. ∎

The decomposition applies to many other sequencing problems with an average completion time objective with only minor modifications. For the single machine scheduling problem of section 3, it becomes even slightly easier since the modification (step 3b) is not needed. The only other difference is the value of γ. For many classic routing and scheduling problems, the algorithm is well-defined given a constant factor β-approximation algorithm and appropriate value γ. It fails, however, for problems where the partition in subproblems or the concatenation is not well defined. For example, when deadlines are involved, then delaying part of the solution (as done in step 3a) may be impossible. However, note that many routing and scheduling problems with hard deadlines have no good approximation algorithms anyway, as checking for a feasible solution may already be NP-complete.

2.1.1. Decomposition analysis. The analysis given here is almost independent of the underlying problem. The proof for Lemma 2.5 is just one line of calculus, but it is a key argument in the analysis. For any $j \in \{1, \ldots, n\}$, denote the index of the subset V_i containing point j by $i(j)$. That means $j \in V_{i(j)}$ or, equivalently, $t_{i(j)} \leq C_j < t_{i(j)+1}$. Lemma 2.5 follows directly from the partition in step 2 and is independent of the underlying problem.

LEMMA 2.5. *$\mathbb{E}[t_{i(j)}] < C_j/a$ for any $j \in \{1, \ldots, n\}$.*

Proof. Note that $t_{i(j)}$ is a stochastic variable of the form $t_{i(j)} = e^{-x}C_j$, where x is uniform on $[0, a]$:

$$\mathbb{E}[t_{i(j)}] = C_j\mathbb{E}[e^{-x}] = \frac{C_j}{a}\int_{x=0}^{x=a} e^{-x}dx = \frac{C_j(1 - e^{-a})}{a} < \frac{C_j}{a}. \tag{2.1}$$

This completes the proof. ∎

We shall prove the following lemma for the TRP, but a similar proof applies to many other problems. Consider an optimal solution σ^* for I, and let C_j^* be the completion time of point j in σ^*. Further, let OPT_i be the optimal value of instance I_i, as defined in step 3a.

LEMMA 2.6. *For any instance I_i, $\text{OPT}_i \leq \sum_{j\in V_i} w_j C_j^* + \gamma t_i \sum_{j\in V_i} w_j$.*

Proof. A feasible solution for I_i is obtained by delaying σ^* by a time γt_i and only visiting points in V_i. ∎

Note that in the proof of Lemma 2.6 we used that delaying the solution is feasible for the TRP. For other sequencing problems, this may not be valid, for example when deadlines are involved. The next lemma follows immediately from the inequalities of Lemmas 2.5 and 2.6.

LEMMA 2.7. *$\mathbb{E}[\sum_i \text{OPT}_i] \leq (1 + \epsilon)\text{OPT}$.*

Proof. From Lemma 2.6,

$$\sum_i \text{OPT}_i \leq \sum_i \sum_{j\in V_i} w_j C_j^* + \gamma \sum_i \sum_{j\in V_i} w_j t_i = \text{OPT} + \gamma \sum_j w_j t_{i(j)}.$$

Next, we take the expected value over the random b and use Lemma 2.5:

$$\mathbb{E}\left[\sum_i \textsc{Opt}_i\right] \leq \textsc{Opt} + \gamma \sum_j w_j \mathbb{E}[t_{i(j)}] \leq \textsc{Opt} + \frac{\gamma}{a}\sum_j w_j C_j$$
$$= \textsc{Opt} + \frac{\epsilon}{\beta}\sum_j w_j C_j \leq (1+\epsilon)\textsc{Opt}.$$

This completes the proof. ∎

Summarizing, we obtained the following decomposition theorem for the weighted, metric TRP. In section 2.2, we show how to reduce further to polynomially bounded weights.

THEOREM 2.8. *For any weighted TRP instance I and constant $\epsilon > 0$, we can find in polynomial time a (random) partition $V_1, V_2, \ldots, V_q$ of V and (random) time points $L_1, L_2, \ldots, L_{q+1}$ such that the following three properties hold. Let I_i be the L_i-delayed instance on points V_i:*

(i) *L_{i+1}/L_i is a constant of the order $e^{O(1/\epsilon^2)}$.*
(ii) *Any solution for I_i can be modified in polynomial time, and with loss of a factor $(1+\epsilon)$ in the value, such that it completes before time L_{i+1}.*
(iii) *If, for all $i \in \{1, \ldots, q\}$, σ_i is an α-approximate solution for I_i that completes before time L_{i+1}, then concatenating the solutions σ_i in the order $i = 1, 2, \ldots, q$ gives a $(1+\epsilon)\alpha$-approximate solution for I in expectation.*

Proof. The partition and the time points $L_i = \gamma t_i$ are defined as in Algorithm DECOMPOSE. Note that $L_{i+1}/L_i = \gamma t_{i+1}/(\gamma t_i) = e^a = e^{\beta\gamma/\epsilon} = e^{O(1/\epsilon^2)}$.

By Lemma 2.4, the modification increases the value by at most a factor $(1+\epsilon)$. Concatenation of the tours σ_i gives a feasible solution for I of expected value at most $\alpha\mathbb{E}[\sum_i \textsc{Opt}_i]$. By Lemma 2.7, this is at most $(1+\epsilon)\alpha\textsc{Opt}$. ∎

Derandomization of the decomposition can be done by enumerating over at most a polynomial number of partitions. For ease of analysis, we treat any two partitions of V that differ only in the labeling as different. For example, $V_1 = \{1\}, V_2 = \emptyset$ and $V_1 = \emptyset, V_2 = \{1\}$ are considered different. Note that q may be much larger than n but is still polynomially bounded since $\log(C_n)$ is polynomially bounded. Let Q be an upper bound on q. Given the values $C_1, \ldots, C_n$, there are no more than $O(nQ)$ possible partitions $V_1, \ldots, V_Q$. This can be seen by letting variable b increase from 0 to a and observing the changes. For any of these partitions, we let b be the smallest value in $[0, a]$ that defines the partition and run the algorithm for that value.

2.2. Reducing the subproblems. Given a TRP instance I on point set $V \cup \{r\}$, we know by Theorem 2.8 how to reduce it to at most n instances I_i with the loss of a factor $(1+\epsilon)$ in the approximation ratio, where each I_i is an L_i-delayed TRP instance with point set $V_i \subseteq V$ and origin r. We shall set $w_v = 0$ for any $v \notin V_i$. Moreover, I_i has the property that there exists a solution of length at most δL_i, where δ is a constant depending on ϵ. For ease of analysis, we will restrict ourselves from now on to those instances in general and shall ignore the reduction that leads to these instances. In particular, we forget about the ϵ of the previous section and let δ be an arbitrary constant.

DEFINITION 2.9. *Let $\delta > 0$ be a given constant. We say that a weighted TRP instance on a metric space with points $V \cup \{r\}$ is* δ-bounded *if*

(i) *it is L-delayed for some L given as part of the input, and*

(ii) *there exists a path of length at most δL that starts in r and visits all points in U, where $U = \{v \in V \mid w_v > 0\}$.*

Note that any delayed instance is δ-bounded for some δ, so the definition is only useful when we consider a class of δ-bounded instances for fixed δ. We will show next how to reduce the problem on weighted δ-bounded instances to the problem on weighted δ-bounded instances where all weights are polynomially bounded (Theorem 2.11).

Bounding weights polynomially. As mentioned in the introduction, one cannot just scale and round the weights of a TRP instance a priori. An interesting corollary of our decomposition is that for δ-bounded instances one can polynomially bound the weights as stated in the next lemma. Subsequently, one might reduce to unweighted instances by replacing any point of weight w by w copies of the same point. The careful reader may note that the reduction below even works when δ is only polynomially bounded in n. The property that δ is a constant will be fully exploited in sections 2.3 and 2.4.

LEMMA 2.10. *For any δ-bounded instance I of weighted TRP and $\epsilon > 0$, we can change the weights ($w_j \to w'_j$) such that all w'_j are nonnegative integers of value at most W' with $W' = O_\epsilon(n^2)$ (where $n = |V|$) and any α-approximate solution σ for the rounded instance I' is a $(1+\epsilon)\alpha$-approximate solution for I.*

Proof. Let $W = \max_j w_j$ be the maximum weight. Round weights as follows:

$$w'_j = \lfloor w_j / M \rfloor, \text{ where } M = \epsilon W/(n + n^2\delta).$$

Then $w'_j \le \lfloor W/M \rfloor = O_\epsilon(n^2)$. Let $\textsc{Opt}'$ be the optimal value of the rounded instance. Then

$$M\textsc{Opt}' \le \textsc{Opt}.$$

Given an α-approximate solution for the rounded instance, we use it as a solution for the original instance. Let C_j be the completion time of point j in the solution found:

$$\sum_j w_j C_j \le M \sum_j (w'_j + 1) C_j \le M\alpha\textsc{Opt}' + M \sum_j C_j \le \alpha\textsc{Opt} + M \sum_j C_j. \tag{2.2}$$

Note that δL is an upper bound on the maximum distance between any two points since the instance is δ-bounded. Hence, $C_j \le L + n\delta L = (1 + n\delta)L$ for any j. Hence,

$$M \sum_j C_j \le Mn(1 + n\delta)L = \epsilon W L \le \epsilon\textsc{Opt}.$$

From (2.2), we obtain that $\sum w_j C_j \le \alpha\textsc{Opt} + \epsilon\textsc{Opt} \le (1+\epsilon)\alpha\textsc{Opt}$. ∎

The lemma above is used as follows. Given a weighted TRP instance and $\epsilon > 0$, we apply Algorithm DECOMPOSE. For each subproblem, we round the weights and apply an $\alpha(1+\epsilon)$-approximation to the rounded instances and use this as a solution for the unrounded subproblem. The rounding will only cost an extra factor $(1+\epsilon)$, so by Theorem 2.8, this yields an α-approximation scheme for the weighted TRP.

THEOREM 2.11. *If for some class of metric spaces and every constant $\delta > 0$ we have an α-approximation scheme for the set of δ-bounded instances with polynomially bounded weights, then we can find an α-approximation scheme for the weighted TRP in the same class of metric spaces.*

The theorem above together with Lemma 2.14 below is our starting point for the following sections. A weaker theorem that we shall not use, but which is of independent interest, results from the following observation: A delayed TRP instance is just a standard TRP instance with an additive nL in the objective. Hence, any α-approximation scheme for the TRP with polynomially bounded weights is also an α-approximation scheme for δ-*bounded* instances with polynomially bounded weights. This leads to the following theorem.

THEOREM 2.12. *If for some class of metric spaces we have an α-approximation scheme for the weighted TRP with polynomially bounded weights, then we can find a polynomial time α-approximation scheme for the weighted TRP (with arbitrary weights) in the same class of metric spaces.*

Segmented TRP. In [11], the authors note that any solution Π for the unweighted TRP can be replaced by a concatenation of $\kappa = O(\log n/\epsilon)$ TSP paths with only a $(1+\epsilon)$ factor increase in value. That means the solution can be partitioned into κ segments such that replacing each segment S by a shortest path that visits the same points as S and has the same start and endpoint as S increases the value of the solution by at most a factor $(1+\epsilon)$. Hence, for finding good approximations, one may consider what we define here as the *κ-segmented version* of the TRP.

DEFINITION 2.13. *In the κ-segmented version of a TRP problem, a solution for a given instance consists of a feasible tour plus numbers $0 = t^{(0)} \le t^{(1)} \le t^{(2)} \le \cdots \le t^{(\kappa)}$, where $t^{(\kappa)}$ is (at least) the maximum completion time in the tour. The* rounded *completion time of a point v is then defined as the smallest value $t^{(i)}$ greater than or equal to the completion time of v. The objective is to minimize the (weighted) sum of rounded completion times.*

The part of the tour between $t^{(i-1)}$ and $t^{(i)}$ is referred to as the ith segment ($i \ge 1$). Note that for $\kappa = 1$, the κ-segmented TRP is equal to the TSP-path problem (with one fixed endpoint r), and $\kappa = n$ corresponds to the standard TRP problem. Stated like this, the authors of [11] show that an unweighted TRP can be reduced to the κ-*segmented* TRP with $\kappa = O(\log n/\epsilon)$. The next lemma shows that for bounded instances we can do much better.

LEMMA 2.14. *There is a $\kappa = O_\epsilon(\log(1+\delta))$ such that the following holds. Let I be a δ-bounded instance, and let I' be the κ-segmented version of I. Then any α-approximate solution for I' is a $(1+\epsilon)^2\alpha$-approximate solution for instance I.*

Proof. By Lemma 2.3, there is a $(1+\epsilon)$-approximate solution for I that completes before time $T = O_\epsilon(1+\delta)L$, where L is the given delay of I. Now consider time points $t^{(h)} = (1+\epsilon)^h L$ for $h = 1, 2, \ldots, \kappa$, where $(1+\epsilon)^\kappa L \ge T$. Then $\kappa = O_\epsilon(\log(1+\delta))$. Rounding completion times to the next value $t^{(h)}$ increases the value by at most another factor $(1+\epsilon)$. Hence, $\textsc{Opt}(I') \le (1+\epsilon)^2\textsc{Opt}(I)$. For any α-approximate solution for I', its total sum of unrounded completion times is at most its total sum of rounded completion times, which is at most $\alpha\textsc{Opt}(I') \le (1+\epsilon)^2\alpha\textsc{Opt}(I)$. ∎

By now, we reduced the subproblems that resulted from the decomposition into δ-bounded instances of the κ-segmented TRP with polynomially bounded weights, where both δ and κ are constants depending on ϵ. To simplify the analysis in the next subsections, where we only consider the subproblems, we shall (as we did before) forget this dependency of δ and κ on ϵ. The final running time of the algorithm is $O(n^{\text{poly}(1/\epsilon)})$.

For the approximation of the subproblems for TRP on trees and the Euclidean plane, in the next sections we can do without the restriction to δ-bounded instances.

(But do note that we already used this property several times in the analysis.) That means we give approximation schemes for the κ-segmented TRP with polynomially bounded weights on trees and in the Euclidean plane. Hence, we give a slightly more general result than needed. (See also the discussion above Theorem 2.11.) For planar graphs, however, the δ-boundedness property is crucial in the analysis.

2.3. TRP on trees. Here we present a PTAS for the weighted TRP on trees. From the discussion above, we know that it is sufficient to give a PTAS for the κ-segmented TRP with polynomially bounded weights, where κ is a given constant. Hence, an instance is given by a constant κ and an edge-weighted tree where for each vertex v an integer weight w_v is given. A solution is given by a tour that visits all points v (with $w_v > 0$) plus numbers $0 = t^{(0)} \le t^{(1)} \le \cdots \le t^{(\kappa)}$. The value of the solution is then defined as in Definition 2.13; i.e., completion times are rounded up to the next value $t^{(i)}$.

Let $V \cup \{r\}$ be the vertices of the tree, and let E be its set of edges. Each edge $e \in E$ is given an integer distance d_e, which we allow to be zero. The distance $d_{v,w}$ between points v, w of the tree is the distance of the path between v and w in the tree. We assume that $w_v \in \{0, 1\}$ since any vertex of weight w_v can be replaced by w_v vertices at zero distance. Let $U = \{v \in V \mid w_v = 1\}$, and denote $|U| = n$. Clearly, we may assume without loss of generality that all leaves of the tree are in $U \cup \{r\}$ and that there are no vertices of degree 2 that are not in $U \cup \{r\}$ since such vertices can be removed by contracting the adjacent edges. Hence, we may assume that the tree has at most $2n + 1$ vertices.

We give a PTAS for the κ-segmented version of the tree TRP described above. Consider such an instance, and let Opt be its optimal value. The algorithm has two steps, rounding the instance and dynamic programming, just as for the QPTAS from [11]. We do provide all the details here in order to convince the reader that κ being a constant does bring the running time down to a polynomial.

2.3.1. Rounding distances. Since all vertex weights are 0 and 1, rounding distances is straightforward. Given $\epsilon > 0$, let $D = \epsilon \max_e\{d_e\}/(2n^3)$ and define for each $e \in E$

$$d'_e = \lfloor d_e/D \rfloor.$$

Then the maximum edge length is $2n^3/\epsilon$. By dynamic programming, we will compute an optimal solution σ' for the rounded instance and use that as a solution for the original instance. For any feasible tour σ, let $Z(\sigma)$ and $Z'(\sigma)$ be its value for, respectively, the unrounded and rounded instances.

Lemma 2.15. *Let σ' be an optimal solution for the rounded instance. Then its total completion time for the unrounded instance is $Z(\sigma') \le (1 + \epsilon)$Opt.*

Proof. First, observe that

$$Dd'_e \le d_e \le D(d'_e + 1). \tag{2.3}$$

Any path between two vertices in the tree has at most $|V| \le 2n$ edges. Hence, in the solution, the path from the root to any vertex has at most $2n^2$ edges and the value of any solution is the sum of at most $2n^3$ edge lengths. By the second inequality of (2.3), this yields

$$Z(\sigma') \le D(Z'(\sigma') + 2n^3). \tag{2.4}$$

Let σ^* be an optimal tour for the unrounded instance. Since σ' is optimal for the rounded instance, we have $Z'(\sigma') \leq Z'(\sigma^*)$. Together with the first inequality of (2.3) this yields

$$DZ'(\sigma') \leq DZ'(\sigma^*) \leq Z(\sigma^*) = \textsc{Opt}. \tag{2.5}$$

From (2.4) and (2.5), we conclude that $Z(\sigma') \leq \textsc{Opt} + D2n^3 \leq \textsc{Opt} + \epsilon \max_e\{d_e\} \leq (1+\epsilon)\textsc{Opt}$. ∎

2.3.2. Dynamic programming. The QPTAS by Arora and Karakostas was built on the observation that for the unweighted TRP one can restrict oneself to κ-segmented instances, where $\kappa = O_\epsilon(\log n)$. We shall argue below that the running time of the dynamic program (DP) can be made polynomial when κ is a constant. To simplify the description of the DP, we first turn the tree into a binary tree rooted at r such that only its leaves need to be visited. This can be done with only a constant factor increase in the number of vertices by adding edges of length zero. In an optimal solution, each edge is traversed at most κ times in each direction since each of the κ segments is a TSP path (a shortest walk on a set of points with a given start and endpoint). Hence, for any vertex v, the part of the solution that is in the subtree rooted at v is composed of at most κ subtours rooted at v, where we define a subtour routed at v as a maximal part of the solution that start and ends in v and only visits vertices in the subtree rooted at v. (The addition of *maximal* is important for the precise description of the DP below.)

In the DP, we store for each node v a set $\mathcal{C}_v$ of possible *crossing configurations*. Each configuration $C \in \mathcal{C}_v$ only gives the start and end times of each subtour rooted at v. The *value* of a configuration is the minimum total completion time of the points from U in the subtree rooted at v, given the configuration C. Note that the size of the DP table is polynomially bounded since the length of the optimal tour is polynomially bounded and κ is a constant.

For a leaf v, we store exactly one configuration for each $t \in \{0, 1, \ldots, T\}$, where T is an upper bound on the tour length. Any such configuration has exactly one subtour which has start time t, end time t, and value t. For any internal node v, we can compute $\mathcal{C}_v$ as follows. For any pair of configurations C_1 and C_2 of its children v_1 and v_2, we compute the concatenated configuration C for v (by just ordering according to crossing times times in C_1 and C_2), which is either infeasible or uniquely defined (assuming that the complete tour is traversed at unit speed). The value of C is the sum of the values of C_1 and C_2 (since all internal nodes have zero weight). If different pairs lead to the same configuration, then we keep the combination with smallest value.

Here is a small example: Let $\kappa = 2$, and assume all edges have length 1. Consider a pair of configurations where for v_1 the pairs of start and end times of the two subtours are given by $C_1 = \{(5, 10), (30, 38)\}$ and where the configuration for v_2 has only one subtour with start and end time $C_2 = \{(12, 20)\}$. Then the resulting configuration for v is $C_v = \{(4, 21), (29, 39)\}$, which is valid given that C_1 and C_2 are valid. If in C_2 we would have 11 instead of 12, then the combination is infeasible since the tour cannot be in v_1 at time 10 and in v_2 at time 11. Also, if the combined configuration has more than κ subtours (which is the case when 12 is replaced by 14, for example), then it is discarded from the DP.

For the root, we select the configuration with the smallest value (which will only have one subtour) and an optimal tour is found by backtracking.

2.4. Euclidean TRP. Arora and Karakostas [11] gave an approximation scheme for the (unweighted) TRP in the Euclidean plane (see also [7]) with a quasi-polynomial running time: $n^{O(\log n/\epsilon^2)}$. The algorithm in [11] is based on the refined PTAS for the Euclidean TSP [6], which is more efficient than the simpler version that was published earlier [5]. In the latter paper, it was shown that there is a $(1+\epsilon)$-approximate TSP tour that crosses the boundary of each square in the quadtree only $O(\log n/\epsilon)$ times. In the refined PTAS, it was proven that $O(1/\epsilon)$ crossings are enough.

The QPTAS [11] contains many details, but intuitively it does follow easily from the TSP-PTAS [6] and the next properties and assumptions: (i) the instance is unweighted; (ii) all relevant distances are polynomially bounded integers; (iii) the solution we search for is composed of $O(\log n/\epsilon)$ TSP paths; and (iv) there are only $O(1/\epsilon)$ crossings with a square of the quadtree per TSP path. Hence, for a given square one can afford to guess basically all possible crossing configurations and still end up with a quasi-polynomial running time.

We reduced the weighted TRP to δ-bounded instances of a κ-segmented TRP with polynomially bounded weights where δ and κ are constant (Lemma 2.14). We shall give a PTAS for a κ-segmented TRP in the Euclidean plane and shall not use the δ-bounded property in the proof. (See the remark above section 2.3.) We first give a high-level description of the TSP algorithm and then focus on the differences. Hence, we assume that the reader is somewhat familiar with the PTAS for the Euclidean TSP. We refer the reader to the survey [7] for omitted details.

2.4.1. The TSP-PTAS in a nutshell.

The quadtree. The first step is to take the smallest square (the *bounding box*) containing all points and to define an $N \times N$ grid on top of it where N is some polynomially bounded, integer power of 2. Next, move each input point to the middle of its grid cell. (Multiple points might move to the same midpoint.) Then take $a, b \in \{1, \ldots, N\}$ uniformly at random and place a $2N \times 2N$ square B (the *enclosing box*) on top of the bounding box, where a and b are the horizontal and vertical offsets. Next, construct the so-called *quadtree*. The root of the tree is square B. Its children are the four $N \times N$ squares obtained when cutting B in four equal sized squares. The same applies to each vertex of the tree except for the leaves which are the 1×1 grid cells.

The subproblems. A subproblem in the DP consists of a vertex of the quadtree (i.e., a square in the plane) plus *crossing information* which states exactly where the tour enters and leaves the square and how these points of entering and leaving are paired. We shall refer to a path inside the square that starts and ends at the boundary of the square as a *fragment* of the tour. The *value* of a subproblem is the minimum total length of a set of fragments that together visit all the points inside the square and that satisfy the crossing information; i.e., the fragments should enter and leave the square as described. To compute the value of one subproblem, we can enumerate over all consistent combinations of subproblems of its four children. Consistent means that the crossing information of the four children should match with each other and with the parent. Consistency is easy to check.

The size of the DP table. The number of vertices in the quadtree is polynomially bounded since the dimension of the grid is polynomially bounded. To bound the number of subproblems of each vertex in the quadtree, a restriction is put on the places where the tour is allowed to cross grid lines. Given the random placement of B, a set of points (called *portals*) on the grid lines is defined such that each square in the quadtree only has $O_\epsilon(\log n)$ of these portals on its boundary. (Although the precise location of portals is crucial, it is not necessary to specify this here since for the

TRP we will keep the portals exactly the same.) A solution is called *portal respecting* if it only crosses grid lines at portals and crosses each portal at most twice. In the DP, we compute the optimal portal respecting solution.[1] With this restriction, there is only a polynomial number of subproblems for each vertex of the quadtree. To see this, let's copy each portal such that now each portal can be crossed at most once. Imagine that we start at a corner of the square and walk around its boundary and for each portal we store it if it is crossed and, if so, if this is the first or second time we encounter this fragment of the tour on our walk around the boundary. Then for each portal this gives three options. Hence, this crossing information is polynomially bounded. The important observation is that this information is enough to define all the pairings as well. This follows easily from the fact that an optimal tour does not cross itself. (See [5] for details.) Hence, the number of subproblems for each vertex of the quadtree is polynomially bounded.

Computation time. To compute the value of one subproblem, we simply enumerate over all combinations of subproblems of its four children. The four subproblems need to be consistent. That means the locations of entering and leaving should match. The value of the subproblem is the smallest sum of the values of four consistent subproblems for its four children.

The loss in the approximation ratio. At two places we lose in the approximation: when moving points to the middle of grid cells and when restricting to portal respecting tours. Clearly, by making the grid dense enough we do not lose too much by moving the input points. The nontrivial part is in the place of the portals. The crucial observation (the so-called structure theorem) made by Arora is that the increase in optimal tour length by making it portal respecting is only a factor $(1 + \epsilon)$ in expectation. More precisely, given any line segment of length L between two input points, the expected extra length that is needed to let it cross grid lines only at portals is $O(\epsilon L)$. (The expectation is over the random choices of the offset values a and b of the enclosing box.) It follows from this property that we can restrict the DP to solutions that only cross grid lines at portals. Further, it is easy to see that if a portal is crossed more than twice, then we can swap parts of the tour at no extra cost and with no additional crossings of grid lines anywhere such that it crosses that portal at most twice. Hence, we may restrict the DP to portal respecting tours. Exactly the same PTAS applies to the TSP-path problem [7].

2.4.2. TRP-PTAS. Any optimal solution for a κ-segmented TRP is the concatenation of at most κ TSP paths. Hence, to cut a long story short, the TRP-PTAS is basically the TSP-PTAS with a larger size of the DP table. Since κ is constant, the running time stays polynomial.

There are only two differences in the algorithm. First, the grid needs to be made denser to keep the error of moving points to the middle of grid cells small. Since vertex weights are polynomially bounded, the dimension of the grid stays polynomially bounded. (See [11].) Given the grid, the construction of the random quadtree and portals remains exactly the same. The property cited above tells us that for each of the at most κ TSP paths, the expected increase in length that is needed to make it portal respecting is at most a factor $(1 + \epsilon)$. Hence, the objective value of the segmented TRP increases by at most a factor $(1 + \epsilon)$ in expectation when we restrict ourselves to solutions that only cross grid lines at portals and cross each portal at

[1] In fact, the structure theorem in [6] states that $O(1/\epsilon)$ crossings with the boundary are sufficient. For our TRP-PTAS, we do not need this and rely on the structure theorem with $O_\epsilon(\log n)$ crossings, which has a much easier proof.

most 2κ times. The main difference is in the DP. Before starting the DP, we guess the length of each of the κ segments. This then already defines the completion times of the segments. We can do this since κ is a constant.

Let's now take 2κ copies of each portal such that each portal is crossed at most once. Then, as in the TSP, for each portal we store it if it is crossed and, if so, if this is the first or second time we encounter this fragment (the path inside the square) while walking around the boundary of the square. In addition, we store

(i) for each crossing the segment that this crossing is on,
(ii) for each crossing whether it goes into or out of the square, and
(iii) for each of the κ segments the length of the part of the segment inside the square.

For (i) and (ii), this information is clearly polynomially bounded. The same holds for (iii) since these distances can be taken as polynomially bounded integers. Further, we know from the TSP that this is enough to encode the pairing within each of the segments since the noncrossing property holds within each of the κ segments. There are at most $\kappa - 1$ fragments that have their two endpoints in different segments (i.e., the time of entering the square is in segment i and the time of leaving the square is in segment $k > i$), and we can guess these pairings explicitly. Finally, for each of these entries in the DP we store its value, defined as the minimum total weighted completion time of the points inside the square given the crossing information of the square. Remember that if a point is on segment i, then its completion time is defined as the completion time of segment i.

The DP is now straightforward. Each leaf of the quadtree has at most one point (after moving points to the middle of grid cells), and for each subproblem on that leaf we shall put the point (if any) on the segment i for the smallest possible i under the restriction of the boundary information. This defines the value of the subproblem; i.e, if it can be fit on segment i, then its completion time is the completion time of segment i, which we guessed before starting the DP.

For any subproblem on a nonleaf vertex, we minimize over all combinations of subproblems of its four children. The four subproblems need to be consistent. That means the crossing information should match on the boundaries of the $1 + 4$ squares and the combinations of the fragments of the four smaller squares should be valid, i.e., no subtours, and the lengths (iii) should match.

The DP just described computes an optimal, portal respecting solution for a κ-segmented TRP, given the lengths of each of the segments. The DP is applied for all possible combinations of segment lengths, and this number of combinations is polynomially bounded since the number of segments, κ, is polynomially bounded.

2.5. Weighted planar graphs. In the TRP on planar graphs, the points that need to be visited are the vertices of an edge-weighted planar graph. In [11], the authors remark that their QPTAS for the Euclidean TRP carries over directly to weighted planar graphs by using the PTAS for the TSP on weighted planar graphs by Arora et al. [8]. This claim turned out to be incorrect [22]. For the TSP-PTAS, the separator is a Jordan curve which divides the graph into an exterior and interior part. The number of portals is $m = O(\log n/\epsilon^2)$, and each portal is crossed at most twice. Hence, the situation appears similar to the Euclidean case. However, in the planar case, the graph is reduced in each dissection step by contracting some of the edges. Uncontracting the edges increases the length of the tour by at most a factor $(1 + \epsilon)$. This is fine for the TSP but is problematic for the TRP: Uncontracting edges early in the TRP path may cause a large increase in the total completion time. This issue was overlooked in [11] and in the conference version of this paper [30]. We show here that a more involved argument using the δ-boundedness and a result by Klein [23]

resolves the issue.

Planar TSP. First, we sketch the steps in the planar TSP-PTAS of [8]. Given an edge-weighted planar graph $G = (V, E)$, a spanning subgraph $G' = (V, E')$ is computed for which (i) the distance between any two points is increased by at most a factor $(1+\epsilon)$, and (ii) the total edge weight of G' is $O(\textsc{Opt}/\epsilon)$, where $\textsc{Opt}$ is the weight of an optimal TSP tour in G. Next, a hierarchical decomposition of G' is constructed, similar to the quadtree for the plane. The separator in this case is a Jordan curve that cuts the graph in an inner and outer part, both with at least a constant fraction of the vertices. Hence, the depth of the recursion is $O(\log n)$. Moreover, for any k, one can contract some of the edges such that after contraction, each of the separating curves cuts the graph in at most k vertices and the weight of the contracted edges in each separator is $O(1/k)$ times the weight of the subgraph that it separates. Over all separators, this adds up to $O((\log n)/k)$ times the weight of G'. Now one may choose $k = \Theta((\log n)/\epsilon^2)$ such that the total weight of the contracted edges is at most $\epsilon\textsc{Opt}$.

An optimal TSP tour in the contracted spanner can be computed by dynamic programming similar to the approach for the Euclidean plane. The subproblem is to find a minimum length set of paths that visits all points in the subgraph and satisfies the crossing information. The crossing information is similar to the Euclidean case except that now the portals are the k vertices on the separator. Uncontracting the edges only adds a factor $(1 + \epsilon)$ to the length of the solution found.

Planar TRP. For the TRP, we act as follows. We know from Lemma 2.14 that we only need to construct a PTAS for a δ-bounded, κ-segmented TRP with polynomially bounded weights, where δ and κ are constants. Unlike our algorithm for the tree and the plane, we shall use the δ-bounded property explicitly here. For our subproblems, only a subset of the vertices needs to be visited due to the decomposition and rounding vertex weights to zero. Hence, let $V \cup \{r\}$ be the vertices of the graph and $U \subseteq V$ be the vertices to be visited. Each edge e is given an integer distance d_e which we allow to be zero. Note that rounding distances can be done easily like we did for the tree-TRP since the vertex weights are polynomially bounded. We skip the precise computation. We may further assume that vertex weights are in $\{0, 1\}$ since any vertex of weight w_v can be replaced by w_v vertices at zero distance.

Since the shortest tour on U may be much smaller than the shortest tour on V, the spanner used for the planar-TSP as described above does not suffice. Fortunately, Klein [23] showed how for given $U \subseteq V$ to construct a subgraph G' that is a $(1 + \epsilon)$-spanner for $U \cup \{r\}$ with total edge length $O(\textsc{Opt}'/\epsilon^4)$, where $\textsc{Opt}'$ is the length of an optimal TSP tour on $U \cup \{r\}$ in G (which may be much smaller than the optimum on $V \cup \{r\}$.) Hence, that will be the spanner we use.

Contracting and uncontracting edges is problematic for the approach suggested in [11] since uncontracting edges early in the TRP path may cause a large increase in the total completion time. We now use that our problem is δ-bounded. There is a tour on $U \cup \{r\}$ of length $O(L)$, where L is the delay. Hence, the total edge length of the spanner is only $O(L/\epsilon^4)$. We apply the construction for the separating curves as used for the TSP which gives an increase in length of $O((\log n)/k)$ times the weight of the spanner (where $n = |V|$), which is $O(L/\epsilon^4)$ as just mentioned. Hence, the total increase in length is $O(L \log n/(\epsilon^4 k))$. We can make this increase at most ϵL by choosing the number of vertices on the separator to be $k = \Theta((\log n)/\epsilon^5)$. Consequently, since any completion time is at least L, uncontracting increases the completion time of each vertex by at most a factor $(1 + \epsilon)$.

An optimal TRP tour in the contracted spanner can be computed by dynamic programming in a way similar to what was done for the plane. The boundary infor-

mation is exactly the same, but the portals are now the k vertices of the contracted graph on the separator. The running time of the DP is $n^{O_\epsilon(\kappa)}$.

2.6. Generalizations and variants. The approximation schemes apply as well in the presence of release times. Here we only briefly state the intuition for this statement. First, observe that the decomposition algorithm and the analysis go through without modification. To see this, note that a constant factor approximation algorithm for the TRP with release times follows easily from the study of the online TRP in which the requests (nodes) arrive over time [25]. Further, note that in the algorithm and the analysis, solutions are only delayed and never moved forward in time. Also for the subproblems, release times give no complications: if release times are rounded to powers of $(1+\epsilon)$, then there is only a constant number of different release times and that can easily be handled by the DP.

Another extension is the *k-repairman problem* in which one needs to find k repairman paths that together visit all points. A constant factor approximation is given in [17]. Also here, the decomposition goes through without modifications. For the subproblems, the PTAS applies as well if k is a constant.

In the *randomized search ratio* problem (see [24]), one has to find a (random) path starting from the root r and visiting all points, and the goal is to minimize $\max_v \mathbb{E}[C(v)]/d(r,v)$, where $d(r,v)$ is the distance from r to v. In [11], the authors mention that if the minimum latency problem has a PTAS for a certain class of metrics, then the randomized search ratio problem has an approximation scheme for that same class of metrics. Thus, our PTAS implies a PTAS for the randomized search ratio for trees, planar graphs, and the Euclidean plane.

There are several related problems for which no PTAS is known yet. One of them is finding a metric embedding on a line such that the average distortion is minimized [16]. The authors of [16] use ideas of the QPTAS for the TRP to obtain a QPTAS for the average distortion problem. It is not clear whether our ideas can be used to obtain a PTAS for metric line embedding as well. Another interesting open problem is the k-TSP problem in edge-weighted planar graphs. A PTAS is known for trees and the Euclidean plane, but the planar case is still open.

The best approximation ratio for the TRP in general is still 3.59 [14]. We showed that if for every constant $\kappa \geq 1$ we have an α-approximation (scheme) for a κ-segmented TRP, then we obtain an α-approximation scheme for the TRP. In our analysis for the different metric spaces, it is crucial that κ is a constant. Intuitively, this property can be exploited in the general case as well to improve on the factor 3.59. A first step for studying a segmented TRP would be the case $\kappa = 2$. Here, for a given TRP instance, we need to return a path visiting all points and one number $i \leq n$. The value of the solution is $iC_i + (n-i)C_n$, where C_i (C_n) is the ith (nth) smallest completion time.

We showed how any constant factor β-approximation algorithm for the TRP together with a PTAS for the δ-bounded subproblem results in a PTAS for the TRP. One may wonder what happens for nonconstant β. Note that $\delta = e^{O_\epsilon(\beta)}$. So if, for example, $\beta = \log\log n$, then $\delta = (\log n)^{O_\epsilon(1)}$ and our DPs for the δ-bounded subproblems only run in quasi-polynomial time.

3. Single machine scheduling under precedence constraints. One of the most intriguing scheduling problems is minimizing total weighted completion times on a single machine under precedence constraints, $1|prec|\sum_j w_j C_j$, in the notation by Graham et al. [21]. The problem is known to be $\mathcal{NP}$-hard [26, 27], and several 2-approximation algorithms are known. Exact polynomial time algorithms are

known for some special cases, e.g., for series parallel posets [26]. Surprisingly, the interval ordered precedence constraint is not one of these. Woeginger [35] gave a 1.62-approximation algorithm, and a 3/2-approximation was given by Ambühl et al. [2]. The same paper shows that scheduling interval orders is in fact $\mathcal{NP}$-hard. Here we give a PTAS for interval ordered precedence constraints. It is interesting to note that the analysis here holds for any type of precedence constraints with the exception of Lemma 3.6. We give a formal definition of interval orders just before that lemma.

An instance of the scheduling problem is given by a set $\mathcal{J}$ of n jobs (labeled $1, 2, \ldots, n$) to be processed on a single machine that can process at most one job at a time. Each job j has a nonnegative integer processing time $p_j \geq 0$ and weight w_j. A partial order $\prec$ on the jobs defines the precedence constraints between jobs. That means that if $j_1 \prec j_2$, then job j_1 must be completed before j_2 can start. The goal is to find a nonpreemptive schedule that minimizes $\sum_{j=1}^{n} w_j C_j$, where C_j is the completion time of job j. We assume that there are no jobs of zero processing time which are not to be preceded by any other job since such jobs can always be completed at time 0.

We apply the same decomposition technique that we used for the TRP; i.e., we reduce to unweighted, δ-bounded instances. For bounded instances, we can partition the time horizon in segments and round the completion time of each job to the end of the segment. The analysis of the decomposition is a bit easier than what we did for the TRP since any solution of an instance has the same length and there is no need for the modification as done in step 3b. On the other hand, for the TRP the approximation of the subproblems was a (relatively simple) extension of the (complex) PTAS for the TSP and we only sketched the main ideas. For the scheduling problem, however, we do not rely on a known PTAS and we give a complete proof of our algorithm for the subproblems.

3.1. A decomposition theorem.

DEFINITION 3.1. *We say that an instance I is L-delayed if it has the additional restriction that no job is allowed to start before some time $L > 0$.*

Algorithm DECOMPOSE (Scheduling):

1) **Approximate:** Apply any constant factor β-approximation algorithm for the problem. Relabel the jobs such that $1 \leq C_1 \leq \cdots \leq C_n$, where C_j is the completion time of job j in the solution ($j \in \{1, 2, \ldots, n\}$).
2) **Partition:** Let $a = \beta\gamma/\epsilon$, and take b uniformly at random from $[0, a]$, where we now take $\gamma = 2$.
   As before, let $t_i = e^{a(i-2)+b}$ for $i = 1, 2, \ldots, q+1$, where q is large enough such that $C_n < t_{q+1}$. Partition the set of jobs $\mathcal{J}$ into sets $\mathcal{J}_i = \{j | t_i \leq C_j < t_{i+1}\}$, $i \in \{1, 2, \ldots, q\}$.
3) **Approximate subproblems:** Let I_i be the instance defined by jobs in $\mathcal{J}_i$ and with an additional delay of $L_i = \gamma t_i$. For each i, find an α-approximation σ_i.
4) **Concatenate:** Return the concatenation of $\sigma_1, \ldots, \sigma_q$ as the final schedule.

Note that the partition in step 2 is well-defined since $t_1 = e^{a-b} \leq e^0 = 1$ and $C_j \geq 1$ for all j. Also note that if $j \prec k$, then $C_j \leq C_k$ in step 1, which implies that precedence constraints are satisfied in the final schedule.

As before, we assume $\epsilon \leq 1$. The next lemma shows that concatenation does not increase completion times.

LEMMA 3.2. *If $\gamma = 2$, then any schedule for I_i is contained in the interval $[2t_i, 2t_{i+1}]$.*

Proof. Since $\beta \geq 1$ and $\epsilon \leq 1$, by assumption we have that $e^a = e^{\beta\gamma/\epsilon} \geq e^2 > 2$. Hence, $t_{i+1} > 2t_i$. Any schedule for I_i completes the latest at time $2t_i + P(\mathcal{J}_i) \leq 2t_i + t_{i+1} < e^a t_i + t_{i+1} = 2t_{i+1}$. Note that γ can be taken smaller than 2 but the value $\gamma = 2$ is good enough. ∎

3.1.1. Analysis. The analysis is almost the same as for the TRP. Lemmas 2.5, 2.6, and 2.7 follow immediately, but for completeness we repeat the conclusions here. For any $j \in [n]$, let $i(j)$ be such that $j \in \mathcal{J}_{i(j)}$, which means $t_{i(j)} \leq C_j < t_{i(j)+1}$. Lemma 2.5 states that

$$\mathbb{E}[t_{i(j)}] < C_j/a. \tag{3.1}$$

Now consider an optimal solution σ^* for I and let C_j^* be the completion time of job j in σ^*. Further, let OPT_i be the optimal value of instance I_i. A feasible solution for I_i could be obtained by delaying σ^* by a time γt_i and removing all jobs not in $\mathcal{J}_i$. Hence,

$$\text{OPT}_i \leq \sum_{j \in \mathcal{J}_i} w_j C_j^* + \gamma t_i \sum_{j \in \mathcal{J}_i} w_j. \tag{3.2}$$

Given (3.1) and (3.2), we follow exactly the proof of Lemma 2.7 and get that

$$\mathbb{E}\left[\sum_i \text{OPT}_i\right] \leq (1+\epsilon)\text{OPT}. \tag{3.3}$$

THEOREM 3.3. *For any instance I of $1|prec|\sum_j w_j C_j$ and constant $\epsilon > 0$, we can find in polynomial time a (random) partition $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_q$ of $\mathcal{J}$ and (random) time points $L_1, L_2, \ldots, L_{q+1}$ such that the following three properties hold. Let I_i be the L_i-delayed instance for job set $\mathcal{J}_i$:*

(i) *L_{i+1}/L_i is a constant of the order $e^{O(1/\epsilon)}$.*
(ii) *In any schedule for I_i, all jobs are placed in the interval $[L_i, L_{i+1}]$.*
(iii) *If, for all $i \in \{1, \ldots, q\}$, σ_i is an α-approximate schedule for instance I_i, then concatenating the schedules σ_i in the order $i = 1, \ldots, q$ gives an $\alpha(1+\epsilon)$-approximate solution for I.*

Proof. The partition and the time points $L_i = \gamma t_i$ are defined as in Algorithm DECOMPOSE. Note that $L_{i+1}/L_i = \gamma t_{i+1}/(\gamma t_i) = e^a = e^{\beta\gamma/\epsilon} = e^{O(1/\epsilon)}$.

Concatenation of the schedules σ_i gives a feasible schedule for I of expected value at most $\alpha\mathbb{E}[\sum_i \text{OPT}_i]$. By (3.3), this is at most $(1+\epsilon)\alpha\text{OPT}$. ∎

3.2. A PTAS for bounded instances. Here we give a $(1+\epsilon)$-approximation for the instances I_i as defined by the decomposition algorithm. Together with Theorem 3.3 this gives a PTAS for interval orders. We do not adopt all specifications of I_i but define the set of δ-bounded scheduling instances, like what was done for the TRP, and give a PTAS for this class of instances.

DEFINITION 3.4. *Let $\delta > 1$ be a given constant. We say that an instance of our scheduling problem is* δ-bounded *if*

(i) *it is L-delayed for some $L > 0$ given as part of the input, and*
(ii) *$p(J) \leq (\delta - 1)L$, i.e., any schedule completes before time δL.*

Let $\mathcal{I}_\delta$ be the set of all δ-bounded instances. Clearly, $I_i \in \mathcal{I}_\delta$ for $\delta = L_{i+1}/L_i = e^{O(1/\epsilon)}$. We shall drop the index i for now and consider arbitrary instances $I_\delta \in \mathcal{I}_\delta$, where δ is a constant. Next, we show that for bounded instances, weights and processing times can be polynomially bounded. Given an instance $I_\delta \in \mathcal{I}_\delta$ with delay L, the interval $[L, \delta L]$ is partitioned into $\kappa = O((\log \delta)/\epsilon)$ time slots with starting times that geometrically increase by a factor $(1+\epsilon)$. We fix an imaginary optimal schedule σ^*, and for each job j we guess a set S_j of consecutive slots with the following properties: (P1) j completes in σ^* in some slot in S_j, and (P2) if $j_1 \prec j_2$, then $\max(S_{j_1}) \leqslant \min(S_{j_2})$. We will assign each job j to some slot in S_j. We guess the slots of large jobs and assign the remaining small jobs in a greedy way. By the second property, we then only have to deal with precedence constraints within a slot. But this will be no issue since the ratio of the end time and start times of each slot is only $(1+\epsilon)$ and we shall overpack each slot by at most a factor $(1+\epsilon)$. Hence, we are basically left with a packing problem.

3.2.1. Rounding weights and processing times. A theorem by Woeginger [35] states that for the problem with arbitrary precedence constraints one may restrict the approximation analysis to the case $1 \leqslant w_j \leqslant n^2$ and $1 \leqslant p_j \leqslant n^2$, where n is the number of jobs. In fact, the proof of that theorem also applies to the special case of interval orders since its proof only reverses the precedence constraints and since the reverse of an interval order is again an interval order (see [35]). Given our decomposition theorem, there is a much easier way to show that we may restrict ourselves to polynomially bounded weights and processing times for instances in $\mathcal{I}_\delta$. Moreover, it applies to any class of precedence constraints.

The rounding is similar though a bit easier than the rounding for the TRP. To round the weights, note that $w_{\max}(J)L$ (with J the set of jobs) is a lower bound on the optimal value since $L \leq C_j$ for any job j. On the other hand, $C_j \leq \delta L$ for any job j, so by rounding weights down to multiples of $M = \epsilon w_{\max}(J)/(n\delta)$, the total error made is at most $nM\delta L = \epsilon w_{\max}(J)L \leq \epsilon \textsc{Opt}$. Note that the maximum rounded weight is $\lfloor w_{\max}(J)/M \rfloor M = O_\epsilon(n)M$. Since all weights are multiples of M, we can, subsequently, divide all weights by M.

A similar rounding holds for processing times. Round all processing times and L down to multiples of D where $D = \epsilon L/(n+1)$. Then the error in the completion time of any job is no more than $(n+1)D$. Hence, the total error is bounded from above by

$$(n+1)Dw(J) = \epsilon L w(J) \leq \epsilon \textsc{Opt}.$$

The maximum rounded processing time is at most $\lfloor \delta L/D \rfloor D = O_\epsilon(n)D$. Since all processing times are multiples of D, we can subsequently divide all by D. Further, let L' be the rounded delay. For the rounded instance, there is a solution in which all jobs complete before time $\delta L \leq \delta(L'+D) \leq 2\delta L'$ (since $D \leq L'$ follows from $n \geq 0$ and $\epsilon \leq 1$). Hence the rounded instance is in $\mathcal{I}_{2\delta}$. Since the precise value of δ is irrelevant (we only use that δ is a constant depending on ϵ), we shall ignore this factor 2 and consider instances in $\mathcal{I}_\delta$ for which all weights and processing times are integers of value $O_\epsilon(n)$.

3.2.2. Making guesses about the optimal schedule. Let $I_\delta \in \mathcal{I}_\delta$, and assume all weights and processing times are integers of value $O_\epsilon(n)$. By definition, any schedule places all jobs in an interval $[L, \delta L]$. Define time points $t^{(h)} = (1+\epsilon)^h L$ for $h = 0, 1, \ldots, \kappa$, where $(1+\epsilon)^\kappa L \geq \delta L$. Then $\kappa = O((\log \delta)/\epsilon)$. For $h = 1, \ldots, \kappa$, we refer to the half-open interval $(t^{(h-1)}, t^{(h)}]$ as *slot* h.

Let σ^* be an optimal schedule for I_δ, and let OPT be its value. Let $\mathcal{S}$ be the set of all consecutive subsets of $\{1, 2, \ldots, \kappa\}$, i.e., $\mathcal{S} = \{\{g, g+1, \ldots, h\} \mid 1 \leq g \leq h \leq \kappa\}$. For each job j, the algorithm *guesses* a set $S_j = \{\min(S_j), \ldots, \max(S_j)\} \in \mathcal{S}$ with the following two properties:

(P1) Any job j in σ^* completes in some slot in S_j. (It may start in an earlier slot though.)

(P2) If $j_1 \prec j_2$, then $\max(S_{j_1}) \leqslant \min(S_{j_2})$.

So far, we did not assume anything about the precedence constraints. Lemma 3.6 says that, for interval orders, we can guess sets $S_j \in \mathcal{S}$ for $j = 1, 2, \ldots, n$ that satisfy properties (P1) and (P2). By guessing we mean that we can enumerate over a polynomial number of sequences $S_1, \ldots, S_n$, where at least one of these sequences satisfies the two properties. We now give a formal definition of an interval order.

DEFINITION 3.5. *A partial order on a set J is an interval order if there is a function that assigns to each $j \in J$ a closed interval $[x_j, y_j]$ such that $j_1 \prec j_2$ if and only if $x_{j_1} < y_{j_2}$.*

Note that for any interval order there is a corresponding set of intervals $[x_j, y_j]$ for which all $2|J|$ endpoints are different.

LEMMA 3.6. *For interval orders, we can guess sets S_j for $j = 1, 2, \ldots, n$ that satisfy properties* (P1) *and* (P2).

Proof. Let $[x_j, y_j]$ be the interval for job j in the interval order. As noted, we may assume that the $2n$ values x_j, y_j are all different. For any $h \in \{1, \ldots, \kappa\}$, let $J^*(h)$ be the set of jobs that complete in slot h in σ^*. Note that $J^*(h)$ may be empty. For each h, we guess whether $J^*(h)$ is empty and, if it is not empty, we guess the job that has the largest x-value among the jobs in $J^*(h)$. Denote it by j^h. Hence, for any h, we guess at most one job, which bounds the number of guesses by $n^{O(\kappa)} = n^{O_\epsilon(1)}$.

Assume from now on that we guessed correctly. For any j, let h_j^* be the slot in which job j completes in σ^*. Note that, by definition, $h^*_{j^h} = h$. The following three properties of h_j^* are easily verified:

(a) If $j \prec j^h$ for some slot h, then $h_j^* \leqslant h\ (= h^*_{j^h})$.

(b) If $j^h \prec j$ for some slot h, then $h \leqslant h_j^*$.

(c) If for some h the guess was $J^*(h) = \emptyset$ or if $x_{j^h} < x_j$, then $h_j^* \neq h$.

Now we define the sets S_j. For any nonempty set $J^*(h)$, we know that job j^h completes in slot h. Hence, we let $S_{j^h} = \{h\}$ in that case. For any other job j, we argue as follows. Since h_j^* satisfies the conditions (a)–(c), we let $S_j = [\min(S_j), \ldots, \max(S_j)]$, where we take $\min(S_j)$ ($\max(S_j)$) as the minimum (maximum) value h_j^* satisfying the conditions (a)–(c). Now property (P1) follows directly. To prove (P2), assume that $j_1 \prec j_2$. We distinguish three cases:

Case 1. $j_2 = j^h$ for some j^h. It follows from (a) that $\max(S_{j_1}) \leqslant h = \min(S_{j_2})$ since $S_{j_2} = \{h\}$.

Case 2. $j_1 = j^h$ for some j^h. It follows from (b) that $\min(S_{j_2}) \geqslant h = \max(S_{j_1})$ since $S_{j_1} = \{h\}$.

Case 3. Now assume that $j_1 \neq j^h, j_2 \neq j^h$ for any j^h. Let $h = \min(S_{j_2})$. Then by (c), $J^*(h) \neq \emptyset$ and $x_{j_2} < x_{j^h}$. It follows from $j_1 \prec j_2$ that $y_{j_1} < x_{j_2} < x_{j^h}$, Hence, $j_1 \prec j^h$ and then (a) implies $\max(S_{j_1}) \leqslant h = \min(S_{j_2})$. ∎

From now on we assume that we guessed sets S_j satisfying (P1) and (P2) and shall no longer make any assumption on the precedence constraints. Say that a job is *large* if its processing time is at least $\epsilon L/\kappa^2$, and call it *small* otherwise. For each

large job, we guess the slot in which it completes in σ^*. Note that there are only $O_\epsilon(1)$ large jobs. For any set $S \in \mathcal{S}$, let

$$J_S = \{j \mid S_j = S \text{ and } j \text{ is small}\}.$$

For every pair (S, h), with $S \in \mathcal{S}$ and $h \in \{1, \dots, \kappa\}$ we guess the total processing time over all jobs $j \in J_S$ which complete in some slot $i \leq h$ in σ^*. Denote this total processing time by $P_{\sigma^*}(S, h)$. The number of pairs (S, h) is $O(\kappa^3) = O_\epsilon(1)$, and for each pair, the number of possible values $P_{\sigma^*}(S, h)$ is $O_\epsilon(n^2)$ since $p_j = O_\epsilon(n)$. Hence, the total number of guesses to be made is $n^{O_\epsilon(1)}$. Assume from now on that we guessed correctly.

3.2.3. Construction and analysis of the schedule. We will partition the jobs into sets $J(1), \dots, J(\kappa)$ (corresponding to the slots). Each job j will be assigned to some set $J(h)$ with $h \in S_j$. Given the assignment of jobs to sets $J(h)$, we schedule the jobs within a set $J(h)$ in any arbitrary order that satisfies the precedence constraints. The sets $J(1), \dots, J(\kappa)$ are placed one after the other in this order, starting at time L. By property (P2), the resulting schedule σ is guaranteed to be feasible.

Each large job j is assigned to $J(h)$ if we guessed that it completes in slot h in σ^*. Next, we define and analyze the assignment of the small jobs. For each $S \in \mathcal{S}$, place the jobs in J_S in nondecreasing order w_j/p_j and do the following: For $h = 1, 2, \dots, \kappa$, (in this order) assign jobs from J_S in order w_j/p_j to sets $J(h)$ until either the total processing time of jobs from J_S in $J(1) \cup \cdots \cup J(h)$ becomes at least $P_{\sigma^*}(S, h)$ or until all jobs from J_S are assigned. Let $P_\sigma(S, h)$ be the total processing time of jobs from J_S in $J(1) \cup \cdots \cup J(h)$. This greedy assignment of small jobs overpacks by at most one small job, i.e., for all pairs S, h:

$$P_\sigma(S, h) \leqslant P_{\sigma^*}(S, h) + \epsilon L/\kappa^2. \tag{3.4}$$

On the other hand, we do not pack too little:

$$P_\sigma(S, h) \geqslant P_{\sigma^*}(S, h). \tag{3.5}$$

LEMMA 3.7. *Any job in $J(h)$ completes before time $(1 + \epsilon)t^{(h)}$.*

Proof. Let $P_\sigma(h) = \sum_{S \in \mathcal{S}} P_\sigma(S, h)$ be the total processing time of small jobs in $J(1) \cup \cdots \cup J(h)$, and let $P_{\sigma^*}(h) = \sum_{S \in \mathcal{S}} P_{\sigma^*}(S, h)$ be the total processing time of small jobs that complete in one of the first h slots in σ^*. Remember that the number of possible sets S is no more than κ^2:

$$P_\sigma(h) = \sum_{S \in \mathcal{S}} P_\sigma(S, h) \leqslant \sum_{S \in \mathcal{S}} \left(P_{\sigma^*}(S, h) + \epsilon L/\kappa^2\right) = P_{\sigma^*}(h) + \epsilon L. \tag{3.6}$$

Assuming that the large jobs were assigned correctly, we see that the total processing time assigned to $J(1) \cup \cdots \cup J(h)$ is at most ϵL plus the length of the first h slots. Hence, any job in $J(h)$ completes before time

$$t^{(h)} + \epsilon L < (1 + \epsilon)t^{(h)}. \qquad \square$$

LEMMA 3.8. *The constructed schedule is $(1 + \epsilon)^2$-approximation for instance I_δ.*

Proof. Let j be a large job that completes in slot h in σ^* and (given that we guess correctly) is assigned to $J(h)$. Its completion time in σ^* is at least $t^{(h-1)}$, and, by

Lemma 3.7, it completes in σ before time $(1+\epsilon)t^{(h)} = (1+\epsilon)^2 t^{(h-1)}$. Hence, for each large job individually the increase in completion time is bounded by a factor $(1+\epsilon)^2$.

For the small jobs, we argue as follows. Since for any $S \in \mathcal{S}$ we placed the jobs in J_S in nondecreasing order of w_j/p_j, it follows from (3.5) that for any $S \in \mathcal{S}$, the total weight of jobs from J_S in the sets $J(1) \cup \cdots \cup J(h)$ is at least the total weight of jobs from J_S in the first h slots of the optimal schedule. Hence, the total weighted completion time of small jobs in σ is at most the total weighted completion time of small jobs in σ^*, times a factor $(1+\epsilon)$ due to the ordering of jobs within a slot. ∎

REFERENCES

[1] F. Afrati, S. Cosmadakis, C.H. Papadimitriou, G. Papageorgiou, and N. Papakostantinou, *The complexity of the travelling repairman problem*, Rairo Inform. Théor. Appl., 20 (1986), pp. 79–87.

[2] C. Ambühl, M. Mastrolilli, N. Mutsanas, and O. Svensson, *On the approximability of single-machine scheduling with precedence constraints*, Math. Oper. Res. 36 (2011), pp. 653–669.

[3] A. Archer and A. Blasiak, *Improved approximation algorithms for the minimum latency problem via prize-collecting strolls*, in Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms, 2010, pp. 429–447, https://doi.org/10.1137/1.9781611973075.36.

[4] A. Archer, A. Levin, and D. P. Williamson, *A faster, better approximation algorithm for the minimum latency problem*, SIAM J. Comput., 37 (2008), pp. 1472–1498, https://doi.org/10.1137/07068151X.

[5] S. Arora, *Polynomial-time approximation schemes for Euclidean TSP and other geometric problems*, in Proceedings of the 37th Symposium on Foundations of Computer Science, 1996, pp. 2–12.

[6] S. Arora, *Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems*, J. ACM, 45 (1998), pp. 753–782.

[7] S. Arora, *Approximation schemes for NP-hard geometric optimization problems: A survey*, Math. Program., 97 (2003), pp. 43–69.

[8] S. Arora, M. Grigni, D. Karger, P. Klein, and A. Woloszyn, *A polynomial-time approximation scheme for weighted planar graph TSP*, in Proceedings of the Ninth Annual ACM–SIAM Symposium on Discrete Algorithms, 1998, pp. 33–41.

[9] S. Arora and G. Karakostas, *Approximation schemes for minimum latency problems*, in Proceedings of the 31st ACM Symposium on Theory of Computing, 1999, pp. 688–693.

[10] S. Arora and G. Karakostas, *A $2+\epsilon$-approximation for the k-mst problem*, in Proceedings of the 11th Symposium on Discrete Algorithms, 2000, pp. 754–759.

[11] S. Arora and G. Karakostas, *Approximation schemes for minimum latency problems*, SIAM J. Comput., 32 (2003), pp. 1317–1337, https://doi.org/10.1137/S0097539701399654.

[12] N. Bansal and S. Khot, *Inapproximability of hypergraph vertex cover and applications to scheduling problems*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 6198, Springer, Berlin, 2010, pp. 250–261.

[13] A. Blum, P. Chalasani, D. Coppersmith, W. Pulleyblank, P. Raghavan, and M. Sudan, *The minimum latency problem*, in Proceedings of the 26th ACM Symposium on Theory of Computing, 1994, pp. 163–171.

[14] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar, *Paths, trees, and minimum latency tours*, in Proceedings of the 44th Symposium on Foundations of Computer Science, 2003, pp. 36–45.

[15] T. Dewilde, D. Cattrysse, S. Coene, F.C.R. Spieksma, and P. Vansteenwegen, *Heuristics for the traveling repairman problem with profits*, Comput. Oper. Res., 40 (2013), pp. 1700–1707.

[16] K. Dhamdhere, A. Gupta, and R. Ravi, *Approximation algorithms for minimizing average distortion*, Theory Comput. Syst., 39 (2006), pp. 93–111.

[17] J. Fakcharoenphol, C. Harrelson, and S. Rao, *The k-traveling repairman*, in Proceedings of the 14th Symposium on Discrete Algorithms, 2003, pp. 646–654.

[18] U. Feige, L. Lovász, and P. Tetali, *Approximating min sum set cover*, Algorithmica, 40 (2004), 219–234.

[19] A. García, P. Jodrá, and J. Tejel, *A note on the travelling repairman problem*, Networks, 40 (2002), pp. 27–31.

[20] M. X. Goemans and J. Kleinberg, *An improved approximation ratio for the minimum latency problem*, Math. Programming, 82 (1998), pp. 111–124.
[21] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, *Optimization and approximation in deterministic sequencing and scheduling: A survey*, Ann. Discrete Mathe., 5 (1979), pp. 287–326.
[22] G. Karakostas, *private communication*, 2014.
[23] P. N. Klein, *A subset spanner for planar graphs, with application to subset TSP*, in Proceedings of the 38th ACM Symposium on Theory of Computing, 2006, pp. 749–756.
[24] E. Koutsoupias, C.H. Papadimitriou, and M. Yannakakis, *Searching a fixed graph*, in Automata, Languages, and Programming (Paderborn, 1996), Lecture Notes in Comput. Sci. 1099, Springer, Berlin, 1996, pp. 280–289.
[25] S. O. Krumke, W. E. de Paepe, D. Poensgen, and L. Stougie, *News from the online traveling repairman problem*, Theoret. Comput. Sci., 295 (2003), pp. 279–294.
[26] E. L. Lawler, *Sequencing jobs to minimize total weighted completion time subject to precedence constraints*, Ann. Discrete Math., 2 (1978), pp. 75–90.
[27] J. K. Lenstra and A. H. G. Rinnooy Kan, *The complexity of scheduling under precedence constraints*, Oper. Res., 26 (1978), pp. 22–35.
[28] V. Nagarajan and R. Ravi, *The directed minimum latency problem*, in Proceedings of the 11th International Workshop, APPROX 2008, and the 12th International Workshop, RANDOM 2008, on Approximation, Randomization and Combinatorial Optimization, 2008, pp. 193–206.
[29] R. A. Sitters, *The minimum latency problem is NP-hard for weighted trees*, in Integer Programming and Combinatorial Optimization, Lecture Notes in Comput. Sci. 2337, Springer, Berlin, 2002, pp. 230–239.
[30] R. A. Sitters, *Polynomial time approximation schemes for the traveling repairman and other minimum latency problems*, in Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, 2014, pp. 604–616.
[31] R. A. Sitters, *Polynomial Time Approximation Schemes for the Traveling Repairman and Other Minimum Latency Problems*, preprint, http://arxiv.org/abs/1307.4289, 2014.
[32] R. A. Sitters and L. Yang, *A* $(2+\epsilon)$*-approximation for precedence constrained single machine scheduling with release times and total weighted completion time objective*, Oper. Res. Lett., 46 (2018), pp. 438–442.
[33] M. Skutella, *A* 2.542*-approximation for precedence constrained single machine scheduling with release dates and total weighted completion time objective*, Oper. Res. Lett., 44 (2016), pp. 676–679.
[34] J. N. Tsitsiklis, *Special cases of traveling salesman and repairman problems with time windows*, Networks, 22 (1992), pp. 263–282.
[35] G. J. Woeginger, *On the approximability of average completion time scheduling under precedence constraints*, Discrete Appl. Math., 131 (2003), pp. 237–252.