A $(2+\varepsilon)$-approximation algorithm for the general scheduling problem in quasipolynomial time

Alexander Armbruster[*] Lars Rohwedder[†] Andreas Wiese[‡]

Abstract

We study the general scheduling problem (GSP) which generalizes and unifies several well-studied preemptive single-machine scheduling problems, such as weighted flow time, weighted sum of completion time, and minimizing the total weight of tardy jobs. We are given a set of jobs with their processing times and release times and seek to compute a (possibly preemptive) schedule for them on one machine. Each job incurs a cost that depends on its completion time in the computed schedule, as given by a separate job-dependent cost function for each job, and our objective is to minimize the total resulting cost of all jobs. The best known result for GSP is a polynomial time $O(\log\log P)$-approximation algorithm [Bansal and Pruhs, FOCS 2010, SICOMP 2014].

We give a quasi-polynomial time $(2+\varepsilon)$-approximation algorithm for GSP, assuming that the jobs' processing times are quasi-polynomially bounded integers. For the special case of the weighted tardiness objective, we even obtain an improved approximation ratio of $1+\varepsilon$. For this case, no better result had been known than the mentioned $O(\log\log P)$-approximation for the general case of GSP. Our algorithms use a reduction to an auxiliary geometric covering problem. In contrast to a related reduction for the special case of weighted flow time [Rohwedder, Wiese, STOC 2021][Armbruster, Rohwedder, Wiese, STOC 2023] for GSP it seems no longer possible to establish a tree-like structure for the rectangles to guide an algorithm that solves this geometric problem. Despite the lack of structure due to the problem itself, we show that an optimal solution can be transformed into a near-optimal solution that has certain structural properties. Due to those we can guess a substantial part of the solution quickly and partition the remaining problem in an intricate way, such that we can independently solve each part recursively.

[*]Technical University of Munich (alexander.armbruster@tum.de). Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 551896423.

[†]University of Southern Denmark (rohwedder@sdu.dk). Supported by Dutch Research Council (NWO) project "The Twilight Zone of Efficiency: Optimality of Quasi-Polynomial Time Algorithms" [grant number OCEN.W.21.268]

[‡]Technical University of Munich (andreas.wiese@tum.de). Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 551896423.

1 Introduction

We consider the following general and fundamental scheduling problem: we are given one machine and a set of jobs J where each job $j \in J$ is characterized by a processing time $p_j \in \mathbb{N}$ and a release time $r_j \in \mathbb{N}$. We seek to compute a (possibly preemptive) schedule for the jobs in J, i.e., each job $j \in J$ is processed for p_j time units in total such that j is not processed before time r_j and the machine works on at most one job at a time. Ideally, we would like to finish each job as early as possible. To quantify this, it is natural to associate a cost to each job $j \in J$ in the computed schedule which depends on the completion time of j; the later j completes, the higher should be the cost for j.

In practical settings, the given jobs might be highly heterogeneous: some jobs might be very important or urgent and, hence, need to be finished quickly. They may even have a hard deadline before which they must finish in *any* feasible schedule. Other jobs might be less critical and even large delays may be tolerable for them at (almost) no cost. Therefore, it makes sense to allow each job j to have a separate function that defines its cost, depending on its completion time. For each job j we denote this function by $\text{cost}_j : [r_j, \infty) \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ and we assume it (only) to be non-decreasing; given a schedule we denote by C_j the completion time of j which then yields a cost of $\text{cost}_j(C_j)$. We assume that we have access the jobs' cost functions via suitable oracles (similarly as in [10], see Section 2 for details). Overall, we seek to minimize the total resulting cost of our jobs, i.e., we want to minimize $\sum_{j \in J} \text{cost}_j(C_j)$. This defines the General Scheduling Problem (GSP) as introduced by Bansal and Pruhs [10], who gave a polynomial time $O(\log \log P)$-approximation algorithm for it. Here, P denotes the ratio between the largest and the smallest job processing times in the input. It is open whether GSP admits a constant factor approximation algorithm, or even a PTAS. For the approach used in [10] it seems unlikely that it can yield a constant factor approximation, as argued in [11]. There has been no progress for the general case of GSP since the work by Bansal and Pruhs [10].

Instead, most research in the context of GSP has focused on special cases of the problem in which the jobs' cost functions are very structured. For example, in the weighted sum of completion times objective each job j has a weight $w_j > 0$ and we seek to minimize $\sum_{j \in J} w_j C_j$. Hence, $\text{cost}_j(t) = w_j \cdot t$ for each job j. This setting admits a polynomial time approximation scheme (PTAS) as shown by Afrati et al. [1]. Another well-studied special case is the weighted flow time objective [11, 15, 21, 5]. For each job j its flow time in a computed schedule is the time between its release and its completion, i.e., $C_j - r_j$, and we seek to minimize $\sum_{j \in J} w_j (C_j - r_j)$. The difference to the weighted sum of completion times objective is "only" the fixed term $\sum_{j \in J} w_j r_j$ and, therefore, the optimal solutions are the same for both objectives. However, approximation ratios of non-optimal solutions are not preserved and constructing approximation algorithms for the weighted flow time objective is much more challenging. For example, it had been a long-standing open question to find even a constant factor approximation algorithm with polynomial running time (quasi-polynomial time algorithms for bounded input data had been known earlier though [14]). In a breakthrough result Batra, Kumar, and Garg [11] presented an $O(1)$-approximation algorithm with pseudopolynomial running time which was subsequently improved to polynomial time by Feige et al. [15]. Finally, the approximation ratio was improved to $1 + \varepsilon$ by Armbruster, Rohwedder, and Wiese [5].

Intuitively, the mentioned algorithms for weighted flow time work with discretized candidate completion times for the jobs. They crucially use that slight changes in a job's flow time change its cost only by a small factor and that this change is proportional for any two jobs with (almost) the same release times. These properties extend to the objective of minimizing the weighted ℓ_p-norm of the jobs' flow times for $p \geq 1$ [11, 6]. However, they do not extend to arbitrary instances of GSP, since there the cost function cost_j of a job j can increase abruptly by large factors when its flow time only increases marginally. Already the

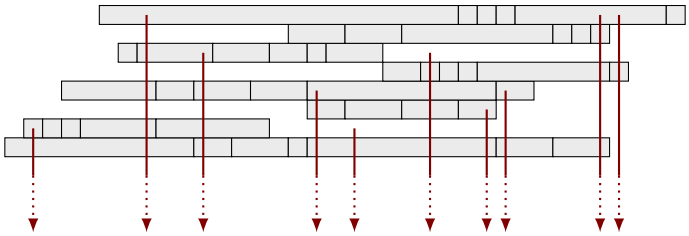

Figure 1: Rectangles and rays in an instance of the rectangle covering problem to which we reduce GSP.

weighted tardiness objective suffers from this issue since a job's cost can stay zero for some time (which may be different for two jobs with the same release time) and only then start to increase. Therefore, it forms an interesting case to study; moreover, it is well-motivated in its own right. Formally, in this objective we are given a due date $d_j \in \mathbb{N}_0$ for each job j which is intuitively a soft deadline for j. Since we might not be able to finish each job j before its due date, for each job j we consider the time between the completion of j and d_j, i.e., $C_j - d_j$, and seek to minimize $\sum_{j \in J} w_j \max\{C_j - d_j, 0\}$. No improvements over the general result of GSP are known for weighted tardiness.

There is another limitation of the mentioned results for special cases of GSP: they assume that *all* jobs' cost functions have the *same* structure. However, they no longer work for heterogeneous sets of jobs, e.g., when each job j either has a hard deadline (the cost is zero if j is completed before the deadline and ∞ otherwise) or its cost in the computed schedule equals its weighted flow time. Even though both cases individually can be approximated well or even solved exactly in polynomial time, this is not clear for the combination of both. This motivates searching for algorithms for the general case of GSP which can thus handle such settings.

1.1 Our contribution

In this paper, we present a quasi-polynomial time $(2+\varepsilon)$-approximation algorithm for (the general case of) GSP, assuming that the jobs' processing times are quasi-polynomially bounded integers. We denote by $p_{\max} := \max_{j \in J} p_j$ the maximum job processing time of a given job.

Theorem 1. *For each $\varepsilon > 0$ there is a $(2+\varepsilon)$-approximation algorithm for the general scheduling problem with a running time of $2^{\mathrm{poly}((1/\varepsilon)^{1/\varepsilon} \log(n+p_{\max}))}$.*

For the special case of weighted tardiness, our approximation ratio improves to $1+\varepsilon$.

Theorem 2. *For each $\varepsilon > 0$ there is a $(1+\varepsilon)$-approximation algorithm for the weighted tardiness problem with a running time of $2^{\mathrm{poly}((1/\varepsilon)^{1/\varepsilon} \log(n+p_{\max}))}$.*

Since weighted tardiness is strongly NP-hard, this is the best possible approximation ratio in quasi-polynomial running time, unless $\mathsf{NP} \subseteq \mathrm{DTIME}(2^{\mathrm{poly}(\log n)})$. Note that our result also implies that the problem cannot be APX-hard (with bounded input data), unless $\mathsf{NP} \subseteq \mathrm{DTIME}(2^{\mathrm{poly}(\log n)})$.

Inspired by the PTAS for weighted flow time [5], we reduce GSP and weighted tardiness to a geometric covering problem in which we need to select a subset of some given axis-parallel rectangles in the plane in order to cover the demand of a given set of rays (see Figure 1). All given rays are oriented vertically downwards. Each rectangle R has a cost and a value that it contributes to satisfy the demands of the rays that it intersects with (in case R is selected). The rectangles are pairwise non-overlapping and organized in

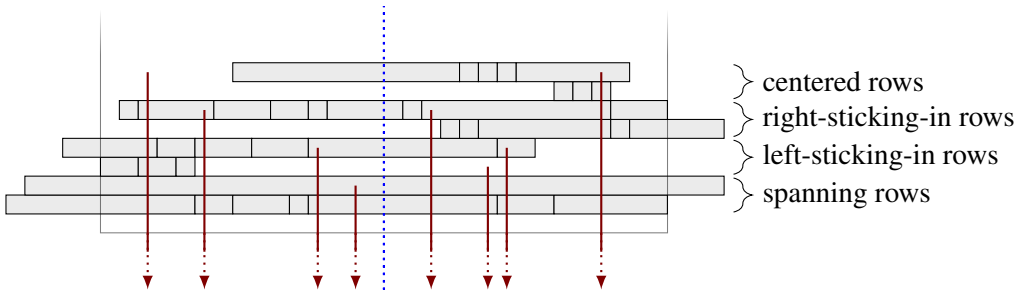

Figure 2: The shaded region indicates a given subproblem. It is split into two parts by the blue dotted line. We lose a factor of 2 for the centered rows crossing this line in this step, but not for the other rows.

rows such that from each row we must select a prefix of its rectangles (or none of them). Our objective is to satisfy the demand of each ray while minimizing the total cost of the selected rectangles. Our reduction loses only a factor of $1 + \varepsilon$ in the objective function value.

Due to the properties of the weighted flow time objective, in the algorithm for that case in [5] it was possible to slightly round the constructed rectangles so that they have a tree-like structure. This structure then guided a dynamic program (DP) for the geometric covering problem. However, in GSP the job's cost function may not have these properties and it seems impossible to ensure a similar structure for the rectangles. Despite this, we construct a quasi-polynomial time algorithm that computes a $(2 + \varepsilon)$-approximation for this geometric covering problem which, due to our reduction, yields a $(2 + \varepsilon)$-approximation for GSP as well. Our strategy is to recursively split the plane vertically in two parts, i.e., into a left and a right subproblem. All rays are vertical and, hence, with respect to them the two subproblems are independent. However, there can be rows whose rectangles intersect with both subproblems (see Figure 2). To compensate for this, we can sacrifice a factor of 2 for the cost of such rows, and give independent copies of its rectangles to each subproblem. More precisely, we can restrict each copy to rectangles contained in the respective subproblem and cut one rectangle (which intersects both subproblems) into two pieces if necessary. One core difficulty is that from each rectangle row we need to select a prefix of its rectangles. Therefore, even if the right subproblem needs to select only one single rectangle from the row, it needs to pay for *all* rectangles in that row, including those that are entirely contained in the left subproblem. Moreover, when we recurse further we cannot sacrifice another factor of 2 in each recursion level; doing so would result in a superconstant approximation ratio. In fact, even if each row contained only one single rectangle it would not be clear how obtain even a constant approximation ratio.

One key observation is that once we sacrificed a factor of 2 for the cost of a row, in each subsequent recursive subproblem the rectangles of the row start on the left of the subproblem, or end on the right of the subproblem, or even both. We call such rows *left-sticking-in* rows, *right-sticking-in* rows, and *spanning* rows, see again Figure 2. We treat these rows differently than the other rows; in this way avoid that in each recursion level we lose another factor of 2 in our approximation ratio for them. Intuitively, for each recursive subproblem we consider an optimal solution for this subproblem. We add certain rectangles to it and in this way, we extend it to a structured near-optimal solution. As a result, for some rays their respective demand is satisfied to a significantly larger extent than necessary, which creates some slack. Also, the solution's structure allows us to guess a potentially very large number of rectangles directly.

If a row with right-sticking-in rectangles is entirely contained in the area of the right subproblem, it is clear that its rectangles should be passed on to the right subproblem. However, a difficult case is when there is a right-sticking-in row from which some rectangles intersect the area of the left subproblem and

some other rectangles intersect the area of the right subproblem, see e.g., the first (upper) right-sticking-in row in Figure 2. To handle this, we argue that we can pass all rectangles of some of these rows to the left subproblem and all rectangles of the remaining rows to the right subproblem. However, if then a row is passed to the *left* subproblem, then we might need its rectangles also to (partially) cover the demand of rays in the *right* subproblem. To ensure this we create some artificial rays for the left subproblem which are, maybe counterintuitively, placed outside the area corresponding to that subproblem. A crucial argument is that it is sufficient to create a polylogarithmic number of these artificial rays due to the mentioned slack. Hence, we can guess them in quasi-polynomial time. We use a similar argumentation for the left-sticking-in rays. However, for those we have the additional difficulty that whenever we select one of its rectangles, we need to pay for all rectangles on the left of it, and such rectangles might not even "belong" to the current subproblem, i.e., they might not even intersect its area. To remedy this, on a high level we argue that we do not lose the mentioned factor of 2 for all rectangles in a row at once in one subproblem. Instead, intuitively we argue that for each rectangle in a row there might be *some* subproblem in which we lose this factor, and for different rectangles these might be different subproblems. In this way, we obtain an approximation ratio of $2 + \varepsilon$ overall.

Finally, for the special case of the weighted tardiness objective, we argue that we can adjust our reduction to the geometric covering problem such that the rectangles' x-coordinates are slightly rounded. This is related to the obtained properties in the corresponding reduction for weighted flow time [21, 5]. However, our obtained properties are much weaker since weighted tardiness is a more general cost function. Nevertheless, we show that even these weaker properties are sufficient to ensure that we do *not* lose a factor of 2 in our approximation ratio as above. On a high level, our rounding ensures that the rectangles' x-coordinates are discretized such that in each recursive subproblem we need to consider only a polylogarithmic number of different *types* for the rows that do not trivially belong to the left or the right subproblem. Even more, for each row type we can directly guess up to a factor of $1 + \varepsilon$ in quasi-polynomial time which of its rectangles are contained in an optimal solution for the subproblem. This makes the resulting algorithm much simpler and, at the same time, improves the approximation ratio to $1 + \varepsilon$.

1.2 Other related work

For GSP, when all release times are equal, Bansal and Pruhs [10] showed that their approach leads to a polynomial time constant approximation and a QPTAS was presented by Antoniadis et al. [3]. Moseley [20] extended the approach to multiple machines (when job migration is allowed) and gave a polynomial time $O(\log \log nP)$-approximation algorithm. Bansal and Batra [8] improved this approximation factor to $O(1)$. For the objective of weighted sum of completion times (and release times) there is even a PTAS for multiple identical machines [1] and there are extensions to uniformly related and unrelated machines [1, 13, 2]. A significant amount of research has focused on online algorithms for weighted flow time. In particular, it is possible to achieve competitive ratios with only a logarithmic dependence on P and other parameters [7, 9]. Another well-studied special case of GSP is the problem of minimizing the total weight of tardy jobs where for each job j we have that $\text{cost}_j(t) = 0$ if $t \leq d_j$ and $\text{cost}_j(t) = w_j$ if $t > d_j$, which generalizes the knapsack problem. Lawler [18] gave a pseudopolynomial exact algorithm for this setting. When all release times are equal, Lawler and Moore [18] gave a PTAS. There have been several recent results that focus on optimizing the (pseudopolynomial) running time for variants of minimizing weight of tardy jobs [12, 17, 16].

2 Algorithmic framework for GSP

Given an instance of GSP, we reduce it to an instance of the rectangle covering problem (RCP) which we define in the following. This reduction loses only a factor of $1+\varepsilon$ in the objective. Then, we present a $(2+\varepsilon)$-approximation algorithm for RCP which yields a $(2+\varepsilon)$-approximation algorithm for GSP.

Formally, RCP is defined as follows. The input consists of a set of axis-parallel rectangles $\mathcal{R}$ and a set of rays $\mathcal{L}$ in the plane with the following properties:

1. Each rectangle $R \in \mathcal{R}$ is of the form $R = [a,b) \times [j, j+1)$ for some values $a, b, j \in \mathbb{N}_0$ and we define $\text{left}(R) := a$ and $\text{right}(R) := b$; it has a given cost $c(R) > 0$ and a given value $p(R) > 0$.
2. The rectangles in $\mathcal{R}$ are pairwise disjoint.
3. The rectangles in $\mathcal{R}$ are partitioned into a set of *rows* $\mathcal{W}$ such that two rectangles $R = [a,b) \times [j, j+1)$, $R' = [a', b') \times [j', j'+1)$ are in the same row if and only if $j = j'$.
4. For each row $W \in \mathcal{W}$ all rectangles in W have the same value, i.e., for any $R, R' \in W$ we have that $p(R) = p(R')$, and the rectangles in W are consecutive along the x-axis, i.e., for each rectangle $R \in W$ there is another rectangle $R' \in W$ with $\text{right}(R) = \text{left}(R')$ or there is no rectangle $R'' \in W$ with $\text{right}(R) < \text{left}(R'')$.
5. Each ray in $\mathcal{L}$ is of the form $\{t\} \times (-\infty, s] =: L(s,t)$ for some $s, t \in \mathbb{N}_0$ and has a given demand $d(L(s,t)) \geq 0$. For a clearer visualization, we draw such a ray as $\{t+1/2\} \times (-\infty, s+1/2]$ since then the drawn ray intersects with the same rectangles as the actual ray but it is not aligned with the rectangle boundaries.

The goal is to compute a set of rectangles $\mathcal{R}' \subseteq \mathcal{R}$ with the following properties:

- For each ray $L(s,t) \in \mathcal{L}$ the rectangles in $\mathcal{R}'$ intersecting $L(s,t)$ cover the whole demand of $L(s,t)$, i.e., $\sum_{R \in \mathcal{R}' : R \cap L(s,t) \neq \emptyset} p(R) \geq d(L(s,t))$.
- From each row $W \in \mathcal{W}$ the set $\mathcal{R}'$ contains a consecutive (possibly empty) set of rectangles starting with the leftmost rectangle, i.e., for each rectangle $R \in W \cap \mathcal{R}'$ the set $\mathcal{R}'$ contains also each rectangle $R' \in W$ with $\text{right}(R') \leq \text{left}(R)$.

Our objective is to minimize the total cost of the rectangles in $\mathcal{R}'$, i.e., to minimize $\sum_{R \in \mathcal{R}'} c(R)$. In our reduction, we will ensure for each row that the cost of all its rectangles together is by at most a bounded factor larger than the cost of any single one of its rectangles. Therefore, in a given instance of RCP we denote by K the maximum ratio of these costs in the same row, i.e., $K := \max_{W \in \mathcal{W}} \frac{\sum_{R \in W} c(R)}{\min_{R \in W} c(R)}$. Note that this implies that each row can have at most K rectangles. For a given instance of RCP we denote by n the number of input bits in binary encoding and, analogously to GSP, we define $p_{\max} := \max_R p(R)$.

We can reduce GSP to RCP while losing only a factor of $1+\varepsilon$ in the approximation ratio. To this end, we will prove the following lemma in Section 3. When measuring the running time to solve a given instance of GSP, we denote by n the number of bits needed to encode the processing time p_j and the release time r_j of each job $j \in J$. For the jobs' cost functions, we assume (similarly as in [10]) that we are given access to an oracle that returns in constant time for each combination of a job $j \in J$ and a value $q > 0$ the earliest time $t \in \mathbb{R}$ such that $\text{cost}_j(t) \geq q$.

Lemma 3. *Given an α-approximation algorithm for RCP with a running time of $f(n, p_{\max}, K)$ there is an $\alpha(1+\varepsilon)$-approximation algorithm for GSP with a running time of $f((n \cdot p_{\max})^{O(1)}, p_{\max}, (1/\varepsilon)^{O(1/\varepsilon^3)})$.*

Due to this reduction, it suffices to construct an algorithm for RCP. We will present a $(2+\varepsilon)$-approximation algorithm for RCP in Section 4 which will prove the following lemma.

Lemma 4. *There is a $(2+\varepsilon)$-approximation algorithm for RCP with a running time of $2^{(1/\varepsilon \cdot K \log n \log p_{\max})^{O(1)}}$.*

Finally, Lemmas 3 and 4 yield Theorem 1. We will present our $(1+\varepsilon)$-approximation algorithm for weighted tardiness (corresponding to Theorem 2) in Section 5.

3 Reduction

In this section we prove Theorem 3. Recall that our goal is to reduce GSP to RCP, losing only a factor of $1+O(\varepsilon)$. This yields a factor of $1+\varepsilon$ by standard rescaling. Throughout the section, we allow our running time to be polynomial in n, the size of the GSP instance and $p_{\max}$.

A standard result in scheduling, see e.g. [19, Chapter 3] is the following condition on when a set of jobs can be scheduled within their deadlines on a single machine.

Lemma 5. *Suppose that for each job $j \in J$ we are given a deadline $d_j \geq r_j$. Then there exists a possibly preemptive schedule on one machine where each job is completed by d_j if and only if*

$$\sum_{j:s \leq r_j < d_j \leq t} p_j \leq t - s \quad \forall s,t \in \mathbb{R} : \min_j r_j \leq s < t \leq \max_j d_j \tag{1}$$

Such a schedule can be found by scheduling the jobs using earliest-deadline-first (EDF). If all input numbers are integral then EDF produces a schedule that completes and preempts jobs only at integral times.

The lemma above implies that to solve GSP, all we need to do is to find job completion times that minimize the cost and satisfy the condition. Then we can find a schedule using EDF on the completion times. Let $T = 2^{k+1}$, where $k \in \mathbb{N}$ with $2^k \leq \max_{j \in J} r_j + \sum_{j \in J} p_j < 2^{k+1}$. Then T is an upper bound on the last completion time in an optimal schedule produced by EDF. We may assume without loss of generality that the smallest release time is zero and the difference between any two consecutive release times $r_j < r_{j'}$ is at most $\sum_j p_j$, since otherwise all jobs released before $r_{j'}$ will in an EDF solution be finished before $r_{j'}$ and therefore the instance would split into two independent instances. Thus, we can bound T by $O(n \sum_j p_j) \leq O(n^2 p_{\max})$.

Based on the considerations above, the following time-indexed integer linear program (ILP1) exactly models GSP:

$$\begin{aligned}
&\min \sum_{j \in J} \sum_{t=r_j}^{T-1} (\mathrm{cost}_j(t+1) - \mathrm{cost}_j(t)) \cdot x_{j,t} \\
\sum_{j:s \leq r_j < t} p_j \cdot x_{j,t} &\geq \sum_{j:s \leq r_j < t} p_j - (t-s) && \forall s,t \in [T] : s < t \\
x_{j,t-1} &\geq x_{j,t} && \forall j \in J, t \in \{r_j+2, r_j+3, \ldots, T\} \\
x_{j,t} &\in \{0,1\} && \forall j \in J, t \in \{r_j+1, r_j+2, \ldots, T\}
\end{aligned}$$

Here $x_{j,t} = 1$ if and only if $C_j > t$. Note that the size of ILP1 is not necessarily polynomial in the size of the GSP instance n, but polynomial in $n + p_{\max}$.

Lemma 6. *Given a solution for GSP with some cost c, in time $\mathrm{poly}(n, p_{\max})$ we can compute a solution for ILP1 with cost at most c. Conversely, given a solution of cost c for ILP1 in time $\mathrm{poly}(n, p_{\max})$ we can compute a solution for GSP of cost at most c.*

Proof. Let C_j, $j \in J$, be the completion times in a solution to GSP. We derive a solution to ILP1 by setting $x_{j,t} = 1$ for all $t < C_j$ and $x_{j,t} = 0$ for all $t \geq C_j$. Then the cost of the solution to ILP1 is

$$\sum_{j \in J} \sum_{t=r_j}^{C_j - 1} \mathrm{cost}_j(t+1) - \mathrm{cost}_j(t) = \sum_{j \in J} \mathrm{cost}_j(C_j)$$

Let $s, t \in [T]$ with $s < t$. Since C_j, $j \in J$, are feasible, we have $\sum_{j : s \leq r_j < C_j \leq t} p_j \leq t - s$. Let $S = \{j \in J : s \leq r_j < t\}$. Then,

$$t - s \geq \sum_{j : s \leq r_j < C_j \leq t} p_j = \sum_{j \in S} p_j - \sum_{j \in S : C_j > t} p_j = \sum_{j \in S} p_j - \sum_{j \in S} p_j \cdot x_{j,t}.$$

By rearranging we obtain that the first constraint of ILP1 is satisfied. The second constraint is satisfied by definition of $x_{j,t}$.

For the other direction, let $x_{j,t}$ be a solution to ILP1. We define $C_j = t$, where t is maximal with $x_{j,t-1} = 1$. Because of the second constraint we then have $x_{j,t} = 1$ if and only if $t < C_j$. Thus,

$$\sum_{j \in J} \mathrm{cost}_j(C_j) = \sum_{j \in J} \sum_{t=r_j}^{C_j - 1} \mathrm{cost}_j(t+1) - \mathrm{cost}_j(t) = \sum_{j \in J} \sum_{t=r_j}^{T-1} (\mathrm{cost}_j(t+1) - \mathrm{cost}_j(t)) \cdot x_{j,t}.$$

Let $s < t$. As above, define $S = \{j \in J : s \leq r_j < t\}$. Then

$$\sum_{j : s \leq r_j < C_j \leq t} p_j = \sum_{j \in S} p_j - \sum_{j \in S : C_j > t} p_j = \sum_{j \in S} p_j - \sum_{j \in S} p_j \cdot x_{j,t} \leq t - s.$$

Thus, EDF applied to the values C_j, $j \in J$, produces a solution to GSP with at most the cost of the solution for ILP1. □

We will now rewrite ILP1 in two steps. Both steps introduce a small approximation error, but this ultimately leads to an integer program that is equivalent to RCP. For each job j we define a sequence of *milestones* $m_0(j), m_1(j), m_2(j), \ldots \in \mathbb{N}$, which are points in time that roughly indicate that the cost function of j increased by a factor of $1+\varepsilon$ compared to the previous milestone. The properties are formalized in the lemma below.

Lemma 7. *For each job j we can in time $\mathrm{poly}(n, p_{\max})$ construct a sequence $m_0(j), m_1(j), m_2(j), \ldots, m_{f_j}(j)$ where*

1. $\mathrm{cost}_j(m_{i+1}(j)) \leq (1+\varepsilon) \cdot \mathrm{cost}_j(m_i(j)+1)$ *for all* $0 \leq i < f_j$,
2. $\mathrm{cost}_j(m_{i+1}(j)+1) > (1+\varepsilon/4) \cdot \mathrm{cost}_j(m_i(j)+1)$ *for all* $0 \leq i < f_j - 1$,
3. $m_0(j) = r_j$,
4. $m_{f_j}(j) = T$.

Proof. The construction is straight-forward: For $j \in J$ set $m_0(j) = r_j$ and for $i > 0$ let $m_i(j) \leq T$ be maximal with

$$\text{cost}_j(m_i(j)) \leq (1+\varepsilon) \cdot \text{cost}_j(m_{i-1}(j)+1).$$

Let f_j be the first index such that $m_{f_j}(j) = T$. Note that the construction satisfies (2) even with ε instead of $\varepsilon/4$[1]. □

The idea of integer program ILP2, which is given below, is that it suffices to restrict ILP1 to the variables $x_{j,m_i(j)}$ (which we denote by $y_{j,i}$ in ILP2) and set all other variables implicitly based on those.

$$\begin{aligned}
\min \sum_{j \in J} \sum_{i=0}^{f_j-1} (\text{cost}_j(m_{i+1}(j)) - \text{cost}_j(m_i(j))) \cdot y_{j,i} & \\
\sum_{\substack{j,i:\ s \leq r_j < t,\\ m_i(j) \leq t < m_{i+1}(j)}} p_j \cdot y_{j,i} \geq \sum_{j:\ s \leq r_j < t} p_j - (t-s) & \qquad \forall s,t \in [T] : s < t \\
y_{j,i-1} \geq y_{j,i} & \qquad \forall j \in J, i \in \{0,1,\ldots,f_j\} \\
y_{j,i} \in \{0,1\} & \qquad \forall j \in J, i \in \{0,1,\ldots,f_j\}
\end{aligned}$$

Lemma 8. *Let $(m_0(j), m_1(j), \ldots, m_{f_j}(j))_{j \in J}$ be milestones satisfying the properties of Theorem 7. Then given a solution of some cost c for ILP1, one can in time $\text{poly}(n, p_{\max})$ compute a solution of cost at most $(1+\varepsilon)c$ for ILP2. Conversely, given a solution of some cost c for ILP2 one can compute in time $\text{poly}(n, p_{\max})$ a solution of the same cost for ILP1.*

Proof. Let $x_{j,t}$ be a solution to ILP1. Define $y_{j,i} = x_{j,m_i(j)}$ for all i, j. Denote by K_j the minimal time such that $y_{j,K_j} = 0$. Then

$$\begin{aligned}
\sum_{j \in J} \sum_{i=0}^{f_j-1} (\text{cost}_j(m_{i+1}(j)) - \text{cost}_j(m_i(j))) \cdot y_{j,i} &\leq \sum_{j \in J} \text{cost}_j(m_{K_j}(j)) \\
&\leq (1+\varepsilon) \sum_{j \in J} \text{cost}_j(m_{K_j-1}(j)+1) \\
&= (1+\varepsilon) \sum_{j \in J} \sum_{t=0}^{m_{K_j-1}(j)} \text{cost}_j(t+1) - \text{cost}_j(t) \\
&\leq (1+\varepsilon) \sum_{j \in J} \sum_{t=0}^{T-1} (\text{cost}_j(t+1) - \text{cost}_j(t)) \cdot x_{j,t}.
\end{aligned}$$

Moreover,

$$\sum_{\substack{j,i:\ s \leq r_j \leq m_0(j) < t,\\ m_i(j) \leq t < m_{i+1}(j)}} p_j \cdot y_{j,i} = \sum_{\substack{j,i:\ s \leq r_j < t,\\ m_i(j) \leq t < m_{i+1}(j)}} p_j \cdot x_{j,m_i(j)} \geq \sum_{j:\ s \leq r_j < t} p_j \cdot x_{j,t} \geq \sum_{j:\ s \leq r_j < t} p_j - (t-s).$$

The prefix constraint holds trivially.

[1] We keep this weaker formulation of the lemma in order to later give an alternative approach in the weighted tardiness objective.

Now consider the other direction. Let $y_{j,i}$ be a solution to ILP2. For all t let $x_{j,t} = y_{j,i}$ where $i \in \{0, 1, \ldots, f_j\}$ with $m_i(j) \leq t < m_{i+1}(j)$. Let K_j be minimal with $y_{j,K_j} = 0$. By definition of $x_{j,t}$, we have that $x_{j,t} = 1$ if and only if $t < m_{K_j}(j)$. Thus,

$$\begin{aligned}\sum_{j \in J} \sum_{t=0}^{T-1} (\mathrm{cost}_j(t+1) - \mathrm{cost}_j(t)) \cdot x_{j,t} &= \sum_{j \in J} \mathrm{cost}_j(m_{K_j}(j)) \\ &= \sum_{j \in J} \sum_{i=0}^{f_j - 1} (\mathrm{cost}_j(m_{i+1}(j)) - \mathrm{cost}_j(m_i(j))) \cdot y_{j,i}\end{aligned}$$

Furthermore,

$$\sum_{j:\ s \leq r_j < t} p_j \cdot x_{j,t} = \sum_{\substack{j,i:\ s \leq r_j < t, \\ m_i(j) \leq t < m_{i+1}(j)}} p_j \cdot y_{j,i} \geq \sum_{j:\ s \leq r_j < t} p_j - (t - s).$$

Again, the other constraint holds trivially. □

Note that the prefix constraint in ILP2, i.e., $y_{j,i-1} \geq y_{j,i}$ for all $j \in J$ and all $i \in \{0, 1, \ldots, f_j\}$, leads to a sequence of variables that for a fixed j show up in the same constraints and thus cannot be chosen independently. In the next transformation, it is our goal to slice this sequence into small, constant length, sequences. We define ILP3($(\tau_k^j)_{j \in J, k \in \mathbb{N}}$) for given increasing sequences $\tau_1^j, \tau_2^j, \tau_3^j, \ldots \in \mathbb{N}$ with $\tau_1^j = 1$. We later construct sequences carefully to make sure that the cost of the solution does not increase significantly. If the sequences are clear from the context, we simply write ILP3.

$$\min \sum_{j \in J} \sum_{k: \tau_k^j \leq f_j} \left[\mathrm{cost}_j(m_{\tau_k^j}(j)) \cdot z_{j,\tau_k^j} + \sum_{i=\tau_k^j}^{\tau_{k+1}^j - 1} (\mathrm{cost}_j(m_{i+1}(j)) - \mathrm{cost}_j(m_i(j))) \cdot z_{j,i} \right]$$

$$\sum_{\substack{j,i: s \leq r_j, \\ m_i(j) \leq t < m_{i+1}(j)}} p_j \cdot z_{j,i} \geq \sum_{j: s \leq r_j < t} p_j - (t - s) \quad \forall s, t \in [T] : s < t$$

$$z_{j,i-1} \geq z_{j,i} \qquad \forall j \in J, k \in \mathbb{N}, \text{ and } i \in \{\tau_k^j + 1, \ldots, \tau_{k+1}^j - 1\} \cap \{0, \ldots, f_j\}$$

$$z_{j,i} \in \{0, 1\} \qquad \forall j \in J, i \in \{0, \ldots, f_j\}$$

Lemma 9. *For any increasing sequences τ_k^j with $\tau_1^j = 1$ and any solution of some cost c for ILP3($(\tau_k^j)_{j \in J, k \in \mathbb{N}}$) one can in time $\mathrm{poly}(n, p_{\max})$ compute a solution for ILP2 of cost at most c.*

Proof. Consider a solution z for ILP3. Define a solution y for ILP2 as follows: for each $j \in J$, let ℓ be maximal with $z_{j,\ell} = 1$. Then set $y_{j,i} = 1$ if $i \leq \ell$ and $y_{j,i} = 0$ otherwise. Since the prefix constraint $y_{j,i-1} \geq y_{j,i}$ is satisfied by definition and since we have $y_{j,i} \geq z_{j,i}$ for all i, j, it is immediate that y is feasible for ILP2. Furthermore, the cost of y in ILP2 is at most the cost of z in ILP3: we consider separately the cost induced by the variables for each job j. Let again ℓ be maximal with $z_{j,\ell} = 1$ and let k be such that $\ell \in \{\tau_k^j, s_k^j + 1, \ldots, \tau_{k+1}^j - 1\}$. The cost of variables $(y_{j,i})_i$ is exactly $\mathrm{cost}_j(m_{\ell+1}(j))$ and the cost of variables $(z_{j,i})_i$ is at least

$$\mathrm{cost}_j(m_{\tau_k^j}(j)) + \sum_{i=\tau_k^j}^{\ell} (\mathrm{cost}_j(m_{i+1}(j)) - \mathrm{cost}_j(m_i(j))) = \mathrm{cost}_j(m_{\ell+1}(j)).$$ □

Let $S \in \{1, 2, \ldots, (1/\varepsilon)^3\}$ be a parameter. We may think of S as an offset selected uniformly at random. Define the sequence $(\tau_k(S))_{k\in\mathbb{N}}$ with $\tau_1(S) = 1$ and $\tau_k(S) = S + (k-1) \cdot (1/\varepsilon)^3$ for $k \geq 2$. We refer to ILP3$((\tau_k(S))_{k\in\mathbb{N}})$ as the ILP that uses $(\tau_k(S))_{k\in\mathbb{N}}$ for each job.

Lemma 10. *For any solution of some cost c for ILP2 there is some value $S \in \{1, 2, \ldots, (1/\varepsilon)^3\}$ such that ILP3$((\tau_k(S))_{k\in\mathbb{N}})$ has a solution of cost at most $(1 + 6\varepsilon)c$.*

Proof. Consider a solution y for ILP2. We define solution $z = y$ for ILP3 and analyze the expected cost with $S \in \{1, 2, \ldots, (1/\varepsilon)^3\}$ chosen uniformly at random. Note that feasibility follows immediately, since ILP3 contains a subset of the constraints of ILP2. Regarding the cost, we will analyze the expected cost for variables of each job $j \in J$ separately. Let ℓ be maximal with $z_{j,\ell} = 1$ and let k be such that $\ell \in \{\tau_k, \tau_k + 1, \ldots, \tau_{k+1} - 1\}$. The cost of $(y_{j,i})_i$ in ILP2 is exactly $\text{cost}_j(m_{\ell+1}(j))$. On the other hand, the cost of the variables $(z_{j,i})_i$ in ILP3 is

$$\sum_{k'=0}^{k} \text{cost}_j(m_{\tau_{k'}(S)}(j)) + \text{cost}_j(m_{\ell+1}(j)). \tag{2}$$

Note that by Theorem 7 we have that $\text{cost}_j(m_i(j) + 1) > (1 + \varepsilon/4)\text{cost}_j(m_{i-1}(j) + 1)$ for all $i > 0$. Furthermore, $\tau_k(S) \geq \tau_{k-1}(S) + (1/\varepsilon)^3$ for $k > 1$. Thus, (2) is at most

$$\begin{aligned}
&\sum_{k'=0}^{k-1} \text{cost}_j(m_{\tau_{k'}(S)}(j)) + \text{cost}_j(m_{\tau_k(S)}(j)) + \text{cost}_j(m_{\ell+1}(j)) \\
&\leq \sum_{k'=0}^{k-1} \text{cost}_j(m_{\tau_{k'}(S)}(j) + 1) + \text{cost}_j(m_{\tau_k(S)}(j)) + \text{cost}_j(m_{\ell+1}(j)) \\
&\leq \text{cost}_j(m_{\tau_{k-1}(S)}(j) + 1) \sum_{h=0}^{\infty} \frac{1}{(1 + \varepsilon/4)^{h(1/\varepsilon)^3}} + \text{cost}_j(m_{\tau_k}(j)) + \text{cost}_j(m_{\ell+1}(j)) \\
&\leq 3\text{cost}_j(m_{\tau_k(S)}(j)) + \text{cost}_j(m_{\ell+1}(j)) \\
&\leq 4\text{cost}_j(m_{\ell+1}(j)).
\end{aligned}$$

Here we assume that ε is a sufficiently small constant. Note that the bound above holds for any choice of S. However, it also looses a factor of 4. We will show that with probability $1 - \varepsilon$ the above inequality holds even with a factor of $1 + 3\varepsilon$. More precisely, we will show that

$$\mathbb{P}[\text{cost}_j(m_{\tau_k(S)}(j)) > \varepsilon \cdot \text{cost}_j(m_{\ell+1}(j))] \leq \varepsilon.$$

Observe that if $k = 1$ then $\text{cost}_j(m_{\tau_k(S)}(j)) = 0 \leq \varepsilon \cdot \text{cost}_j(m_{\ell+1}(j))$. Furthermore, if $\ell \in \{\tau_k + (1/\varepsilon)^2, \tau_k(S) + (1/\varepsilon)^2 + 1, \ldots, \tau_{k+1}(S) - 1\}$ then we have

$$\text{cost}_j(m_{\ell+1}) > (1 + \varepsilon)^{(1/\varepsilon)^2}\text{cost}_j(m_{\tau_k(S)}) > 1/\varepsilon \cdot \text{cost}_j(m_{\tau_k(S)}).$$

Hence, consider the event that $k \in \{\ell - 1, \ell - 2, \ldots, \ell - (1/\varepsilon)^2\}$ and $k > 1$. This can only coincide with at most $(1/\varepsilon)^2$ of the $(1/\varepsilon)^3$ possible values of O. Thus, by uniformly random choice, the probability is at most ε. It follows that the cost of the variables $(z_{j,i})_i$ in ILP3 are in expectation at most

$$\begin{aligned}
\text{cost}_j(m_{\ell+1}(j)) + 3\varepsilon \cdot \text{cost}_j(m_{\ell+1}(j)) + \mathbb{P}[\text{cost}_j(m_{\tau_k(S)}(j)) > \varepsilon \cdot \text{cost}_j(m_{\ell+1}(j))] \cdot 3 \cdot \text{cost}_j(m_{\ell+1}(j)) \\
\leq (1 + 6\varepsilon)\text{cost}_j(m_{\ell+1}(j)). \quad \square
\end{aligned}$$

As a final step in transforming the ILPs, we want to avoid having large jumps in the cost within a group of variables $\{z_{j,i} : i \in \{\tau_k, \tau_k + 1, \ldots, \tau_{k+1} - 1\}\}$ for any job $j \in J$. Formally, we call i a *large jump* for job j, if $\text{cost}_j(m_{i+1}(j)) > 1/\varepsilon \cdot \text{cost}_j(m_i(j))$. We derive the sequence $(\tau_k^j(S))_{k\in\mathbb{N}}$ from $(\tau_k(S))_{k\in\mathbb{N}}$ by inserting each large jump (which is not yet part of the sequence) at the position that maintains the increasing order within the sequence.

Lemma 11. *For any solution of some cost c for ILP3($(\tau_k(S))_{k\in\mathbb{N}}$) there there is a solution to ILP3($(\tau_k^j(S))_{j\in J,k\in\mathbb{N}}$) with cost at most $(1+\varepsilon)c$.*

Proof. Any solution for ILP3($(\tau_k(S))_{k\in\mathbb{N}}$) remains feasible for ILP3($(\tau_k^j(S))_{j\in J,k\in\mathbb{N}}$). Comparing the costs in ILP3($(\tau_k(S))_{k\in\mathbb{N}}$) and ILP3($(\tau_k^j(S))_{j\in J,k\in\mathbb{N}}$), we have that the coefficient in the objective for variables $z_{j,i}$ where i is a large jump can increase from $\text{cost}_j(m_{i+1}(j)) - \text{cost}_j(m_i(j))$ to $\text{cost}_j(m_{i+1}(j))$. Since it is a large jump, this increase is only by a factor of at most $1 + \varepsilon$. Hence, also the total cost of the solution increases at most by a factor of $1 + \varepsilon$. □

We are now ready to prove the main lemma of the reduction.

Proof of Theorem 3. Consider an instance of GSP and let OPT be the value of the optimal solution. From Theorem 6 it follows that there is a solution of cost at most OPT for ILP1. Then, by Theorem 6 there is also a solution of cost at most $(1+\varepsilon)$OPT for ILP2. Because of Theorem 10 and Theorem 11 there is some $S \in \{1, 2, \ldots, (1/\varepsilon)^3\}$ such that ILP($(\tau_k^j(S))_{j\in J,k\in\mathbb{N}}$) has a solution of cost at most $(1 + O(\varepsilon))$OPT. We guess this value of S.

We will now rewrite ILP3 as an instance of the Rectangle Covering Problem (RCP). Assume without loss of generality that the jobs are labelled with $J = \{1, 2, \ldots, n\}$ and are ordered by release time, that is, $r_1 \le r_2 \le \cdots \le r_n$. For each job j and each $k \in \{0, 1, \ldots, f_k\}$ we introduce one row $\sum_{j'=1}^{j-1}(f_j + 1) + k$.

The rectangles of the row corresponding to j, k are defined as follows: there is one rectangle $R(j, i) = [m_i(j), m_{i+1}(j)) \times [\sum_{j'=1}^{j-1}(f_j+1)+k, \sum_{j'=1}^{j-1}(f_j+1)+k+1)$ for every $i \in \{\tau_k^j(S), \tau_k^j(S)+1, \ldots, \tau_{k+1}^j(S)-1\}$. The intuition is that solutions to ILP3 correspond to solutions for RCP, where a rectangle $R(j, i)$ is selected if and only if $z_{j,i} = 1$.

The cost of the rectangle is $c(R(j, i)) = \text{cost}_j(m_{i+1}(j))$ if $i = \tau_k^j(S)$ and $c(R(j, i)) = \text{cost}_j(m_{i+1}(j)) - \text{cost}_j(m_i(j))$ otherwise. Note that this is precisely the coefficient of the variable $z_{j,i}$ in the objective of ILP3. The rays mimic the covering constraint of ILP3: We set $p(R(j, i)) = p_j$. Consider some $s, t \in [T]$ with $s < t$. Let j be minimal with $s \le r_j$. Then we introduce one ray $L(\sum_{j'=1}^{j-1}(f_j+1), t) = [t] \times [\sum_{j'=1}^{j-1}(f_j+1), \infty)$ with demand $d(\sum_{j'=1}^{j-1}(f_j + 1), t) = \sum_{j:s\le r_j<t} p_j - (t - s)$, i.e., the right-hand side of the covering constraint of ILP3. The rectangles that intersect $L(c, t)$ are exactly those $R(j, i)$ for which $s \le r_j$ and $t \in [m_i(j), m_{i+1}(j))$. Note that this corresponds to those $z_{j,i}$ appearing on the left-hand side of the covering constraint. Thus, with the transformation between RCP and ILP3 as noted above, the covering constraint of ILP3 is satisfied if and only if the demands of the rays are satisfied. Since rows correspond to the set of rectangles $\{R(j, i) : i \in \{\tau_k, \ldots, \tau_{k+1} - 1\}\}$ for some j and k and RCP enforces prefixes of rows to be selected, this again corresponds one-to-one to the prefix constraint of ILP3.

Our definition of RCP requires that all costs are strictly positive, which we can easily establish by the following preprocessing. Consider a rectangle of cost zero. If it is the first rectangle of a row, without loss of generality any optimal solution contains it, so we remove it from the instance and decrease the demand of all rays that intersect it by its value. If the rectangle is not the first in its row, without loss of generality any optimal solution, which selects the preceeding rectangle, will also select this rectangle. Hence, we can merge these two rectangles into one with the cost of the first rectangle.

In any row W we have $|W| \leq 2(1/\varepsilon)^3$ rectangles: this is because the difference of any two consecutive $\tau_k(S), \tau_{k+1}(S)$ is by definition at most $2(1/\varepsilon)^3$ and therefore this also holds for any two consecutive $\tau_k^j(S), \tau_{k+1}^j(S)$. Since there are no large jumps within a row, each rectangle must have cost at most $(1/\varepsilon)^{|W|} \cdot c(R_1)$, where R_1 is the first rectangle of W. In order to bound the ratio of costs between any pair of rectangles of one row, we modify the cost of any rectangle $R \in W$ to $c'(R) = c(R) + \varepsilon/|W| \cdot c(R_1)$. Since any solution that selects R must also select R_1, this increases (after applying it to all rows) the total cost by only a factor of $1 + \varepsilon$. The resulting instance satisfies that the ratio between the cost of the entire row to the cost of any rectangles of the same row is at most

$$K \leq |W| \frac{(1/\varepsilon)^{|W|} \cdot c(R_1) + \varepsilon/|W| \cdot c(R_1)}{\varepsilon/|W| \cdot c(R_1)} \leq (1/\varepsilon)^{O(1/\varepsilon^3)}.$$

To summarize, we obtained an instance of RCP, which has a solution of cost at most $(1 + O(\varepsilon))\mathrm{OPT}$ (assuming the correct guess of S). It is easy to see that the size of this instance is $\mathrm{poly}(n, p_{\max})$. Using the given algorithm for RCP, we approximate the instance obtain a solution of cost $\alpha(1 + O(\varepsilon))\mathrm{OPT}$. We then apply Theorem 9 to obtain a solution of the same cost for ILP2, Theorem 8 to obtain a solution of the same cost for ILP1, and finally Theorem 6 to obtain the solution of the same cost for GSP. One can scale ε by a constant to achieve the claimed rate of $\alpha(1 + \varepsilon)$. □

4 Algorithm for RCP

Suppose that we are given an instance of RCP defined by a set of rectangles $\mathcal{R}$ and a set of rays $\mathcal{L}$. Let n be the total size of the input and define $p_{\max} := \max_R p(R)$. First, we argue that it suffices to construct an algorithm in which the number of different rectangle costs appear in the exponent of the running time, since we can round those costs to only $O(\log n)$ pairwise different values.

Lemma 12. *Assume that for any $M \geq 1$ there is an α-approximation algorithm with a running time of $2^{(M \cdot K \log n \log p_{\max} \log \max_R \mathrm{right}(R))^{O(1)}}$ for instances of RCP with at most M different rectangle costs. Then there is an $(1 + \varepsilon)\alpha$-approximation algorithm for arbitrary instances of RCP with a running time of $2^{(1/\varepsilon \cdot K \log n \log p_{\max})^{O(1)}}$.*

Proof. First we argue that we may assume $T = O(n)$. Suppose there exists an interval $I = [a, b)$ with $a, b \in \mathbb{N}_0$ such that

- there is no ray $L(s, t) \in \mathcal{L}$ with $t \in I$,
- there is no rectangle $R \in \mathcal{R}$ with $\mathrm{left}(R) \in I$ or $\mathrm{right}(R) \in I$ and
- there exists a rectangle $\mathcal{R} \in R$ with $\mathrm{right}(R) \geq b$.

Intuitively, we just delete the vertical strip $I \times \mathbb{R}$. Formally, do the following: We decrease each of the following quantities simultaneously by $b - a$:

- t for every ray $L(s, t) \in \mathcal{L}$ where $t \geq b$,
- $\mathrm{left}(R)$ for every rectangle $R \in \mathcal{R}$ with $\mathrm{left}(R) \geq b$, and
- $\mathrm{right}(R)$ for every rectangle $R \in \mathcal{R}$ with $\mathrm{left}(R) \geq b$.

This operation does not change whether a set of rectangles is a prefix of a row nor does it change whether a certain rectangle and a certain ray intersect, so the feasible and optimal solutions do not change. We do this operation repeatedly for an interval I of maximal length. This can only happen $|\mathcal{R}| + |\mathcal{L}|$ times. And after that, we have that for any t with $0 \leq t \leq \max_{R \in \mathcal{R}} \operatorname{right}(R)$, there exists a ray $L(s,t) \in \mathcal{L}$, or there is rectangle $R \in \mathcal{R}$ with $\operatorname{left}(R) = t$ or there exists a rectangle $\mathcal{R} \in R$ with $\operatorname{right}(R) = t$. So $\max_{R \in \mathcal{R}} \operatorname{right}(R) = O(|\mathcal{R}| + |\mathcal{L}|) = O(n)$ and thus also $T = O(n)$.

Now we guess the rectangle $R_{\max}$ with maximal cost in the optimal solution OPT. We discard all rectangles R' with cost larger than $c(R_{\max})$ and all rectangles that are in the same row as such an R' on the right of it. Let $\mathcal{R}_{\text{disc}}$ denote these rectangles. Note that the optimal solution cannot select any rectangle from $\mathcal{R}_{\text{disc}}$, as it has to select a prefix of each row. Next, let $\text{APX}_{\text{cheap}}$ denote all rectangles in rows in which no rectangle has a cost of more that $\varepsilon c(R_{\max})/|\mathcal{R}|$. We select all rectangles in $\text{APX}_{\text{cheap}}$. Note that $c(\text{APX}_{\text{cheap}}) \leq |\text{APX}_{\text{cheap}}| \cdot \varepsilon c(R_{\max})/|\mathcal{R}| \leq \varepsilon c(\text{OPT})$.

Let $\mathcal{R}' = \mathcal{R} \setminus (\mathcal{R}_{\text{disc}} \cup \text{APX}_{\text{cheap}})$. For $R \in \mathcal{R}'$, let $\tilde{c}(R)$ be the smallest power of $1+\varepsilon$ larger than $c(R)$ and let $c'(R) := \lceil \frac{|\mathcal{R}| \cdot K}{\varepsilon^2 c(R_{\max})} \tilde{c}(R) \rceil$. We call the given algorithm with the instance I' consisting of the rectangles $\mathcal{R}'$ with cost c' and the original demand rays $\mathcal{L}$, but a ray $L(s,t) \in \mathcal{L}$ has demand $d'(s,t) = \max\{0, d(s,t) - \sum_{R \in \text{APX}_{\text{cheap}} : R \cap L(s,t) \neq \emptyset} p(R)\}$. Let APX be the obtained solution. Then we output $\text{APX} \cup \text{APX}_{\text{cheap}}$.

Clearly, $\text{APX} \cup \text{APX}_{\text{cheap}}$ is feasible. Next, we analyze the cost. The cost of two rectangles in the same row can differ at most by a factor of K. So for $R \in R'$, we have $c(R) \geq \frac{\varepsilon c(R_{\max})}{K \cdot |\mathcal{R}|}$. Thus $c(R) \leq \frac{\varepsilon^2 c(R_{\max})}{|\mathcal{R}| \cdot K} c'(R)$ and $c'(R) \leq \frac{|\mathcal{R}| \cdot K}{\varepsilon^2 c(R_{\max})} \tilde{c}(R) + 1 \leq \frac{|\mathcal{R}| \cdot K}{\varepsilon^2 c(R_{\max})} (\tilde{c}(R) + \varepsilon c(R)) \leq \frac{|\mathcal{R}| \cdot K}{\varepsilon^2 c(R_{\max})} (1 + 2\varepsilon) c(R)$. As $\text{OPT} \setminus \text{APX}_{\text{cheap}}$ is a feasible solution for I', we have $c'(\text{APX}) \leq \alpha c'(\text{OPT} \setminus \text{APX}_{\text{cheap}}) \leq \alpha \cdot c'(\text{OPT})$. Thus $c(\text{APX} \cup \text{APX}_{\text{cheap}}) \leq c(\text{APX}) + c(\text{APX}_{\text{cheap}}) \leq (1 + 2\varepsilon)\alpha \cdot c(\text{OPT}) \leq (1 + 3\varepsilon)\alpha \cdot c(\text{OPT})$. Rescaling ε by a factor of 3 yields the desired $(1+\varepsilon)\alpha$-approximation algorithm.

Finally, we analyze the running time. There are at most $|\mathcal{R}|$ options for $R_{\max}$. In I', let R, R' be two rectangles in the same row with $c'(R) \leq c'(R')$. Then as they are in the same row, we have $c(R) \geq 1/K \cdot c(R')$ and therefore $c'(R) \geq 1/((1+2\varepsilon)K) \cdot c'(R') \geq 1/(2K) \cdot c'(R')$. Thus the cost of two rectangles in a row differs by at most $2K$ in the instance I'. As for each $R \in \mathcal{R}'$ we have $c(R_{\max}) \geq c(R) \geq \frac{\varepsilon c(R_{\max})}{K \cdot |\mathcal{R}|}$, there are at most $O(\log(K \cdot |\mathcal{R}|/\varepsilon)/\varepsilon)$ different values for $\tilde{c}(R)$ and thus also at most the same number of values for $c'(R)$. This yields the desired bound on the running time. □

Assume that we are given an instance of RCP with at most M different rectangle costs for some value $M \geq 1$ and assume w.l.o.g. that $\min_{R \in \mathcal{R}} \operatorname{left}(R) = 0$. Our algorithm is based on a recursion in which the input of each recursive call consists of

- an instance of RCP defined by a set of rays $\mathcal{L}'$ and a set of rectangles $\mathcal{R}'$; let $\mathcal{W}'$ be the partition of $\mathcal{R}'$ into rows,
- an area A of the form $A = [\operatorname{left}(A), \operatorname{right}(A)) \times [0, \infty)$ for two values $\operatorname{left}(A), \operatorname{right}(A) \in \mathbb{N}_0$ such that $\operatorname{right}(A) - \operatorname{left}(A)$ is a power of 2,
- for each row $W \in \mathcal{W}'$ there is a rectangle $R \in W$ with $W \cap A \neq \emptyset$,
- there are at most $(K \cdot M \cdot \log(p_{\max} \cdot \max_R \operatorname{right}(R))/\varepsilon)^{O(1)}$ rays $L \in \mathcal{L}'$ outside A, i.e., such that $L \cap A = \emptyset$.

Note that there may be rectangles $R \in \mathcal{R}'$ with $R \not\subseteq A$. The recursive call returns a solution to the RCP instance defined by $\mathcal{R}'$ and $\mathcal{L}'$. At the end of our algorithm, we output the solution returned by the (main)

recursive call in which $\mathcal{R}' := \mathcal{R}$, $\mathcal{L}' := \mathcal{L}$, $\text{left}(A) = 0$, and $\text{right}(A) := T$ where T is the smallest power of 2 that is at least $\max_{R \in \mathcal{R}} \text{right}(R)$. Any solution to this subproblem is a solution to our given instance.

Assume we are given a recursive call as defined above. The base case of our recursion arises when $\text{right}(A) - \text{left}(A) = 1$. We can solve such instances by a simple dynamic program in quasi-polynomial time, using that there are only $(K \cdot M \cdot \log(p_{\max} \cdot T)/\varepsilon)^{O(1)}$ rays $L \in \mathcal{L}'$ outside A, i.e., in $\mathcal{L}'_{\text{out}} := \{L \in \mathcal{L}' : L \cap A = \emptyset\}$.

Lemma 13. *If* $\text{right}(A) - \text{left}(A) = 1$*, we can solve the subproblem defined by the recursive call exactly in time* $(n \cdot p_{\max})^{O(|\mathcal{L}'_{\text{out}}|)}$.

Proof. We solve this case via a dynamic program. There is a cell for every combination of a row $W \in \mathcal{W}$, a covered demand $p(s,t) \in \mathbb{N}_0$ with $p(s,t) \leq d(s,t)$ for each $L(s,t) \in \mathcal{L}'_{\text{out}}$ and a covered demand $p(A) \in \mathbb{N}_0$ with $p(A) \leq \max\{d(s,t) : L(s,t) \in \mathcal{L}'_{\text{out}}\}$. The subproblem corresponding to a cell $(W, (p(s,t))_{L(s,t) \in \mathcal{L}'_{\text{out}}}, p(A))$ is to compute a subset S of rectangles in rows W and below of minimal cost with the following properties or decide that such a set does not exist:

- The set S contains a prefix from each row.
- The total demand covered on each ray $L(s,t) \in \mathcal{L}'_{\text{out}}$ at least $p(s,t)$, i.e. $p(\{R \in S : R \cap L(s,t) \neq \emptyset\}) \geq p(s,t)$.
- the total value of rectangles intersecting with A is exactly $p(A)$, i.e. $p(\{R \in S : R \cap A \neq \emptyset\}) = p(A)$
- all rays $L(s,t) \in \mathcal{L}'$ that begin in or below W are covered by S.

Intuitively, we use $p(A)$ to remember the amount covered on any ray that starts above W and intersects A. As they all intersect with the same rectangles from rows W and below, it suffices to remember this number. The dynamic program can be solved as follows: First of all, the subproblem is infeasible if there exists a demand ray $L(s,t) \in \mathcal{L}' \setminus \mathcal{L}'_{\text{out}}$ with $s = \text{left}(A)$, that starts in or below row W and fulfills $d(s,t) > p(A)$. For a subproblem in the bottom row W, we can iterate through all possible prefixes of W, check the demand constraints for each prefix and select the minimal cost prefix if there are multiple ones or declare the subproblem infeasible if no feasible prefix exists. This yields the optimal solution for such a DP-cell.

So suppose that W is not the bottom row. Then we iterate through all possible prefixes $S_W \subseteq W$ of rectangles in row W. Let W' be the row directly below W, i.e., such that there is no row between W and W'. For each prefix, the resulting subproblem corresponds the DP-cell $(W', (p'(s,t))_{L(s,t) \in \mathcal{L}'_{\text{out}}}, p'(A))$ where $p'(s,t) = \max\{0, p(s,t) - p(R)\}$ if there exists a rectangle $R \in S_W$ intersecting with $L(s,t)$ and $p'(s,t) = p(s,t)$ otherwise, and $p'(A) = \max\{0, p(A) - p(R)\}$ if there exists a rectangle $R \in S_W$ intersecting with A and $p'(A) = p(A)$ otherwise. Let S' be the solution for the DP-cell $(W', (p'(s,t))_{L(s,t) \in \mathcal{L}'_{\text{out}}}, p'(A))$. For each prefix $S_W \subseteq W$, we check whether $S_W \cup S'$ is feasible for the current subproblem and select the minimal cost solution among all feasible ones.

Now we show that the algorithm solves the problem optimally. Feasibility follows directly from construction. To show optimality we use induction. We already argued that the base case is solved optimally. Let S be the optimal solution for a cell which is not a base case and let $S_W = S \cap W$. Then $S \setminus S_W$ is also a feasible solution for the subproblem. Let S' be the solution for the subproblem given by the dynamic program. Then $c(S') \leq c(S \setminus S_W)$. So $c(S_W \cup S') \leq c(S)$ and it remains to show that $S_W \cup S'$ is feasible. It certainly contains only prefixes, covers a demand $p(s,t)$ on each ray $L(s,t) \in \mathcal{L}'_{\text{out}}$, covers all rays that start in or below W' and the value of rectangles intersecting A is also $p(A)$. And for a ray $L(s,t) \in \mathcal{L}' \setminus \mathcal{L}'_{\text{out}}$

that start in or below W, but not in or below W', the following happens. The rectangles intersecting $L(s,t)$ (except possible the ones in W) are exactly the rectangles in rows in or below W' that intersecting with A. Hence, $L(s,t)$ is covered to the same extend by S' and $S \setminus S_W$ and thus also to the same extend by $S' \cup S_W$ and S. Thus, $L(s,t)$ is also covered. This shows that the algorithm computes the optimal solution to every subproblem

The output is the minimum cost solution among all values of $p(A)$ of the cell $(W, (p(s,t))_{L(s,t)\in\mathcal{L}'}, p(A))$ where W is the highest row and $p(s,t) = d(s,t)$ for each $L(s,t) \in \mathcal{L}'_{\text{out}}$, which is the optimal solution to the given RCP instance. The running time can be bounded as follows. There are at most n values for W and $n \cdot p_{\max}$ options for each p, this yields a total number of DP-states of $(n \cdot p_{\max})^{O(|\mathcal{L}'_{\text{out}}|)}$. One DP-state for a row W can be solved in time $O(|W|) = O(n)$, yielding the desired running time. □

Assume now that $\text{right}(A) - \text{left}(A) > 1$. We classify the rows $\mathcal{W}'$ into four different types which we will handle differently in the following, see also Figure 2.

Definition 14. *A row $W \in \mathcal{W}'$ is*

- centered *if it is contained in the interior of A, i.e., for each rectangle $R \in W$ we have that* $\text{left}(A) < \text{left}(R)$ *and* $\text{right}(R) < \text{right}(A)$*,*
- right-sticking-in *if each rectangle $R \in W$ satisfies* $\text{left}(A) < \text{left}(R)$ *and there exists a rectangle $R' \in W$ with* $\text{right}(A) \leq \text{right}(R')$*,*
- left-sticking-in *if each rectangle $R \in W$ satisfies* $\text{right}(R) < \text{right}(A)$ *and there exists a rectangle $R' \in W$ with* $\text{left}(R') \leq \text{left}(A)$*,*
- spanning *if there exists a rectangle $R \in W$ with* $\text{left}(R) \leq \text{left}(A)$ *and a rectangle $R' \in W$ with* $\text{right}(A) \leq \text{right}(R')$.

We denote by $\mathcal{W}_{\text{center}}, \mathcal{W}_{\text{right}}, \mathcal{W}_{\text{left}}$, and $\mathcal{W}_{\text{span}}$ the set of centered, right-sticking-in, left-sticking-in, and spanning rows, and by $\mathcal{R}_{\text{center}}, \mathcal{R}_{\text{right}}, \mathcal{R}_{\text{left}}$, and $\mathcal{R}_{\text{span}}$ the union of their rectangles, respectively. Intuitively, we will compute a solution to the given subproblem in which we lose a factor of $2+\varepsilon$ on the cost of the rectangles in centered rows (compared to their cost in the optimal solution) and a factor of $1+\varepsilon$ on the cost of rectangles in right-sticking-in and spanning rows. This has some similarities to the treatment of tasks according to their time windows in [4]. For the rectangles in left-sticking-in rows, this is more complicated and we will lose a factor of $1+\varepsilon$ on the cost of rectangles R with $\text{left}(R) \leq \text{left}(A)$ and a factor of $2+\varepsilon$ on the cost of rectangles R' with $\text{left}(R') > \text{left}(A)$.

We define a reference solution $S \subseteq \mathcal{R}'$ which is a solution to our given subproblem that optimizes an auxiliary cost function c_{APX}, motivated by the factors we lose for the different rectangle types, defined as

$$\begin{aligned} c_{\text{APX}}(S') := 2 \cdot c(S' \cap \mathcal{R}_{\text{center}}) + c(S' \cap \mathcal{R}_{\text{right}}) + c(S' \cap \mathcal{R}_{\text{span}}) + c(R \in S' \cap \mathcal{R}_{\text{left}} : \text{left}(R) \leq \text{left}(A)) \\ + 2 \cdot c(R \in S' \cap \mathcal{R}_{\text{left}} : \text{left}(R) > \text{left}(A)) \end{aligned}$$

for each $S' \subseteq \mathcal{R}'$. Our algorithm will involve guessing certain properties. Intuitively, we will argue later that we compute a solution with small overall cost if we guess these properties corresponding to S.

4.1 Definition of structured near-optimal solution

Based on S we define a more structured near-optimal solution S^+ with $S \subseteq S^+$ and $c(S^+) \leq (1 + O(\varepsilon/\log T))c(S)$. First, we consider all left-sticking-in rows. Let $\text{mid}(A) := (\text{right}(A) - \text{left}(A))/2$. We consider each pair $(c', p') \in \mathbb{N}_0^2$ such that there is a row $W \in \mathcal{W}_{\text{left}}$ for which

- its leftmost rectangle $R \in W$ has cost $c(R) = c'$,
- for each rectangle $R' \in W$ its value $p(R')$ satisfies $p(R') \in [(1+\varepsilon)^{p'}, (1+\varepsilon)^{p'+1})$, and
- there is a rectangle $R'' \in W$ for which $\mathrm{left}(R'') < \mathrm{mid}(A) \leq \mathrm{right}(R'')$.

For each such pair (c', p') let $\mathcal{W}_{\mathrm{left}}^{c',p'}$ denote the set of all rows with the above properties and let $n_{\mathrm{left}}^{c',p'}$ denote the number of rows in $\mathcal{W}_{\mathrm{left}}^{c',p'}$ from which S selects *at least one* rectangle intersecting A. Also, let $\mathcal{R}_{\mathrm{left}}^{c',p'} := \bigcup_{W \in \mathcal{W}_{\mathrm{left}}^{c',p'}} W$ denote the set of all rectangles in these rows. Note that the total cost of these rectangles in S is at least $c' \cdot n_{\mathrm{left}}^{c',p'}$. In S^+ we want to select additional rectangles from certain rows in $\mathcal{W}_{\mathrm{left}}^{c',p'}$. Due to this, we will oversatisfy the demands of some rays in $\mathcal{L}'$. This will create some slack which we will exploit later. If $n_{\mathrm{left}}^{c',p'} < \frac{2K \cdot \log T}{\varepsilon}$ there are only few rows from which S selects a rectangle and we will simply guess these rows and rectangles later. Suppose that $n_{\mathrm{left}}^{c',p'} \geq \frac{2K \cdot \log T}{\varepsilon}$. Let $\mathcal{W}_{\mathrm{left,add}}^{c',p'}$ denote the bottom-most $\lfloor \frac{\varepsilon}{2K \cdot \log T} \cdot n_{\mathrm{left}}^{c',p'} \rfloor$ rows in $\mathcal{W}_{\mathrm{left}}^{c',p'}$ from which S does not contain any rectangle intersecting A; in case there are less than $\lfloor \frac{\varepsilon}{2K \cdot \log T} \cdot n_{\mathrm{left}}^{c',p'} \rfloor$ such rows then we define that $\mathcal{W}_{\mathrm{left,add}}^{c',p'}$ is the set of all these rows. We will add to our solution S^+ *all* rectangles in *all* rows in $\mathcal{W}_{\mathrm{left,add}}^{c',p'}$; we denote them by $\mathcal{R}_{\mathrm{left,add}}^{c',p'}$. Let $W_{\mathrm{left,filled}}^{c',p'}$ denote the top-most row such that for each row $W \in \mathcal{W}_{\mathrm{left}}^{c',p'}$ below it, we have that $S \cup \mathcal{R}_{\mathrm{left,add}}^{c',p'}$ contains a rectangle in row W intersecting A. In particular, for each row in $\mathcal{W}_{\mathrm{left}}^{c',p'}$ underneath $W_{\mathrm{left,filled}}^{c',p'}$ the solution S^+ selects the rectangle intersecting the line $\{\mathrm{left}(A)\} \times \mathbb{R}$. This will make it easy to guess those rectangles since it suffices to guess $W_{\mathrm{left,filled}}^{c',p'}$.

In a similar fashion we select additional rectangles for S^+ from right-sticking-in and spanning rows. The procedure is identical for both cases, we describe it only for right-sticking-in rows. We define $\mathcal{W}_{\mathrm{right}}^{c',p'}$, $n_{\mathrm{right}}^{c',p'}$, $\mathcal{R}_{\mathrm{right}}^{c',p'}$, and the set of considered pairs $(c', p') \in \mathbb{N}_0^2$ in exactly the same way as $\mathcal{W}_{\mathrm{left}}^{c',p'}$, $n_{\mathrm{left}}^{c',p'}$ and $\mathcal{R}_{\mathrm{left}}^{c',p'}$ above. We define $\mathcal{W}_{\mathrm{right,add}}^{c',p'}$ as the bottom-most $\lfloor \frac{\varepsilon}{2K \cdot \log T} \cdot n_{\mathrm{right}}^{c',p'} \rfloor$ rows in $\mathcal{W}_{\mathrm{right}}^{c',p'}$ from which S does not contain any rectangle R with $\mathrm{left}(R) < \mathrm{mid}(A) \leq \mathrm{right}(R)$, and we define $\mathcal{R}_{\mathrm{right,add}}^{c',p'}$ to be their rectangles.

Overall, we define $S^+ := S \cup \bigcup_{c',p'} \mathcal{R}_{\mathrm{left,add}}^{c',p'} \cup \mathcal{R}_{\mathrm{right,add}}^{c',p'} \cup \mathcal{R}_{\mathrm{span,add}}^{c',p'}$.

Lemma 15. *The set S^+ has the following properties:*

- $c(S^+) \leq (1 + \varepsilon/\log T) c(S)$
- *for each pair (c', p') with $n_{\mathrm{left}}^{c',p'} \geq \frac{2K \cdot \log T}{\varepsilon}$ the set S^+ contains the rectangle intersecting the line $\{\mathrm{left}(A)\} \times \mathbb{R}$ from the row $W_{\mathrm{left,filled}}^{c',p'}$ and from each row in $\mathcal{W}_{\mathrm{left}}^{c',p'}$ underneath $W_{\mathrm{left,filled}}^{c',p'}$,*
- *for each pair (c', p') with $n_{\mathrm{right}}^{c',p'} \geq \frac{2K \cdot \log T}{\varepsilon}$ the set S^+ contains the rectangle intersecting the line $\{\mathrm{mid}(A)\} \times \mathbb{R}$ from the row $W_{\mathrm{right,filled}}^{c',p'}$ and from each row in $\mathcal{W}_{\mathrm{right}}^{c',p'}$ underneath $W_{\mathrm{right,filled}}^{c',p'}$,*
- *for each pair (c', p') with $n_{\mathrm{span}}^{c',p'} \geq \frac{2K \cdot \log T}{\varepsilon}$ the set S^+ contains the rectangle intersecting the line $\{\mathrm{mid}(A)\} \times \mathbb{R}$ from the row $W_{\mathrm{span,filled}}^{c',p'}$ and from each row in $\mathcal{W}_{\mathrm{span}}^{c',p'}$ underneath $W_{\mathrm{span,filled}}^{c',p'}$,*
- *for each $\mathrm{set} \in \{\mathrm{left}, \mathrm{right}, \mathrm{span}\}$ and for each pair $(c', p') \in \mathbb{N}_0^2$ for which $n_{\mathrm{set}}^{c',p'} \geq \frac{2K \cdot \log T}{\varepsilon}$ the following holds: if a ray $L(s,t) \in \mathcal{L}'$ intersects with a rectangle from a row $W \in \mathcal{W}_{\mathrm{set}}^{c',p'}$ above $W_{\mathrm{set,filled}}^{c',p'}$ and additionally*

– $\text{left}(A) \leq t < \text{mid}(A)$ *if* $\text{set} = \text{left}$ *and*

– $\text{mid}(A) \leq t < \text{right}(A)$ *if* $\text{set} \in \{\text{right}, \text{span}\}$

we have that

$$p(\{R \in S^+ \cap \mathcal{R}_{\text{set}}^{c',p'} : R \cap L(s,t) \neq \emptyset\}) \geq p(\{R \in S \cap \mathcal{R}_{\text{set}}^{c',p'} : R \cap L(s,t) \neq \emptyset\}) + (1+\varepsilon)^{p'} \lfloor \frac{\varepsilon}{2K \cdot \log T} \cdot n_{\text{set}}^{c',p'} \rfloor.$$

Proof. Consider a group (c', p') and let $\text{set} \in \{\text{left}, \text{right}, \text{span}\}$. Then $c(S \cap \mathcal{R}_{\text{set}}^{c',p'}) \geq c' \cdot n_{\text{set}}^{c',p'}$ as S contains rectangles in $n_{\text{set}}^{c',p'}$ different rows of $\mathcal{W}_{\text{set}}^{c',p'}$ and the first rectangle in each such row has a cost of c'. Recall that $\mathcal{R}_{\text{set,add}}^{c',p'}$ contains all rectangles from up to $\frac{\varepsilon}{2 \cdot K \cdot \log T} n_{\text{set}}^{c',p'}$ rows. Furthermore, the total cost of all rectangles in a row is at most a factor K larger than the cost of a single rectangle and the first rectangle in each row has a cost of c'. Thus, we have that

$$c(\mathcal{R}_{\text{set,add}}^{c',p'}) \leq K \cdot c' \cdot \frac{\varepsilon}{2 \cdot K \cdot \log T} n_{\text{set}}^{c',p'} \leq \frac{\varepsilon}{2 \cdot \log T} \cdot c(S \cap \mathcal{R}_{\text{set}}^{c',p'} W).$$

Using this for all (c', p') and each $\text{set} \in \{\text{left}, \text{span}, \text{right}\}$, we get $c(S^+) \leq (1 + \frac{\varepsilon}{2 \log T}) c(S)$.

From the definition of $W_{\text{set,filled}}^{c',p'}$ we obtain directly that S^+ contains the rectangles from row $W_{\text{set,filled}}^{c',p'}$ and every row in $\mathcal{W}_{\text{set}}^{c',p'}$ below $W_{\text{set,filled}}^{c',p'}$ that intersect with

- the line $\text{left}(A) \times \mathbb{R}$ if $\text{set} = \text{left}$ and
- the line $\text{mid}(A) \times \mathbb{R}$ if $\text{set} \in \{\text{span}, \text{right}\}$.

So it remains to show that S^+ covers substantially more than S. Consider a group (c', p') and suppose that $n_{\text{left}}^{c',p'} \geq \frac{2K \cdot \log T}{\varepsilon}$. Let $L(s,t) \in \mathcal{L}'$ be a ray that intersects a rectangle from a row $W \in \mathcal{W}_{\text{left}}^{c',p'}$ above $W_{\text{left,filled}}^{c',p'}$ and fulfills $\text{left}(A) \leq t < \text{mid}(A)$. Recall that $\mathcal{W}_{\text{left,add}}^{c',p'}$ denotes the bottom-most $\lfloor \frac{\varepsilon}{2K \log T} n_{\text{left}}^{c',p'} \rfloor$ rows in $\mathcal{W}_{\text{left}}^{c',p'}$ from which S does not contain any rectangle intersecting A. Each row $W \in \mathcal{W}_{\text{left,add}}^{c',p'}$ is spanning A_{left}, i.e. there exists a rectangle $R' \in W$ with $\text{left}(R') \leq \text{left}(A)$ and a rectangle $R'' \in W$ with $\text{mid}(A) \leq \text{right}(R'')$. So there also exists a rectangle R with $\text{left}(R) \leq t < \text{right}(R)$. As W is below or equal to $W_{\text{left,filled}}^{c',p'}$ and $L(s,t)$ starts above $W_{\text{left,filled}}^{c',p'}$, the ray $L(s,t)$ intersects with R. So there are $\lfloor \frac{\varepsilon}{2K \log T} n_{\text{left}}^{c',p'} \rfloor$ rectangles in $\mathcal{R}_{\text{left,add}}^{c',p'}$ that intersect with $L(s,t)$. So $p(\{R \in \mathcal{R}_{\text{left,add}}^{c',p'} : R \cap L(s,t) \neq \emptyset\}) \geq (1+\varepsilon)^{p'} \lfloor \frac{\varepsilon}{2K \log T} n_{\text{left}}^{c',p'} \rfloor$. This yields the last property for $\text{set} = \text{left}$.

For $\text{set} \in \{\text{right}, \text{span}\}$ the proof is very similar. Consider a group (c', p') and suppose that $n_{\text{set}}^{c',p'} \geq \frac{2K \cdot \log T}{\varepsilon}$ and $L(s,t) \in \mathcal{L}'$ is a ray that intersects a rectangle from a row $W \in \mathcal{W}_{\text{set}}^{c',p'}$ above $W_{\text{set,filled}}^{c',p'}$ and fulfills $\text{mid}(A) \leq t < \text{right}(A)$. Recall that $\mathcal{W}_{\text{set,add}}^{c',p'}$ denotes the bottom-most $\lfloor \frac{\varepsilon}{2K \log T} n_{\text{set}}^{c',p'} \rfloor$ rows in $\mathcal{W}_{\text{set}}^{c',p'}$ from which S does not contain any rectangle intersecting A_{right}. In each row $W \in \mathcal{W}_{\text{set,add}}^{c',p'}$ there exists a rectangle R with $\text{left}(R) \leq t < \text{right}(R)$. As W is below or equal to $W_{\text{set,filled}}^{c',p'}$ and $L(s,t)$ starts above $W_{\text{set,filled}}^{c',p'}$, the ray $L(s,t)$ intersects with R. So there are $\lfloor \frac{\varepsilon}{2K \log T} n_{\text{set}}^{c',p'} \rfloor$ rectangles in $\mathcal{R}_{\text{set,add}}^{c',p'}$ that intersect with $L(s,t)$, which as before, yields the last property for $\text{set} \in \{\text{right}, \text{span}\}$ and completes the proof. □

4.2 Algorithm

Algorithmically, we guess certain properties of S^+, select some rectangles from $\mathcal{R}'$ accordingly, and then partition the remaining problem into two subproblems which we then solve recursively. First, for each pair (c', p') we guess whether $n_{\text{left}}^{c',p'} < \frac{2K \cdot \log T}{\varepsilon}$. If this is the case, we guess $\mathcal{W}_{\text{left}}^{c',p'}$ and all rectangles in $\mathcal{R}_{\text{left}}^{c',p'}$ that are contained in S^+ and select those. Since $n_{\text{left}}^{c',p'} < \frac{2K \cdot \log T}{\varepsilon}$ we can do this in time $n^{O(K \log T/\varepsilon)}$. Assume now that $n_{\text{left}}^{c',p'} \geq \frac{2K \cdot \log T}{\varepsilon}$. We guess $W_{\text{left,filled}}^{c,p}$ and for each row $W \in \mathcal{W}_{\text{left}}^{c,p}$ underneath $W_{\text{left,filled}}^{c,p}$ we select the rectangle $R \in W$ intersecting $\{\text{left}(A)\} \times \mathbb{R}$ and all rectangles on the left of R. Similarly, for each pair (c', p') we guess whether $n_{\text{right}}^{c',p'} < \frac{2K \cdot \log T}{\varepsilon}$ and if yes, we guess $\mathcal{W}_{\text{right}}^{c',p'}$ and all rectangles in $\mathcal{R}_{\text{right}}^{c',p'}$ that are contained in S^+ and select them. Otherwise, we guess $W_{\text{right,filled}}^{c,p}$ and for each row $W \in \mathcal{W}_{\text{right}}^{c,p}$ underneath $W_{\text{right,filled}}^{c,p}$ we select the rectangle $R \in W$ intersecting $\{\text{mid}(A)\} \times \mathbb{R}$ and all rectangles on the left of R. We handle the spanning rows in the same way as the right-sticking-in rows. Let APX_{mid} denote the selected rectangles.

We want to split the remaining problem into a left and a right subproblem for the areas $A_{\text{left}} := [\text{left}(A), \text{mid}(A)) \times [0, \infty)$ and $A_{\text{right}} := [\text{mid}(A), \text{right}(A)) \times [0, \infty)$ and for sets of rectangles $\mathcal{R}'_{\text{left}}$ and $\mathcal{R}'_{\text{right}}$, respectively, and for certain sets of rays which we will define in the following.

Centered rows. First, we consider the centered rows $\mathcal{W}_{\text{center}}$. For each row $W \in \mathcal{W}_{\text{center}}$ we do the following. Let $W_{\text{left}} \subseteq W$ denote all rectangles $R \in W$ with $R \subseteq A_{\text{left}}$ and let $W_{\text{right}} \subseteq W$ denote all rectangles $R \in W$ with $R \subseteq A_{\text{right}}$. If $W_{\text{left}} = W$ then we simply assign all rectangles in W to $\mathcal{R}'_{\text{left}}$; similarly, if $W_{\text{right}} = W$ then we add all rectangles in W to $\mathcal{R}'_{\text{right}}$. Suppose that $W_{\text{left}} \neq W \neq W_{\text{right}}$. Assume first that there is a rectangle $R_{\text{mid}} \in W$ with $\text{left}(R_{\text{mid}}) < \text{mid}(A) < \text{right}(R_{\text{mid}})$; note that there can be at most one such rectangle. We divide R_{mid} into a left and a right half defined by $R_{\text{mid,left}} := R_{\text{mid}} \cap A_{\text{left}}$ and $R_{\text{mid,right}} := R_{\text{mid}} \cap A_{\text{right}}$. We define $c(R_{\text{mid,left}}) := c(R_{\text{mid}})$ and assign all rectangles in $W_{\text{left}} \cup \{R_{\text{mid,left}}\}$ to $\mathcal{R}'_{\text{left}}$. Also, we define $c(R_{\text{mid,right}}) := \sum_{R \in W_{\text{left}}} c(R) + c(R_{\text{mid}})$. The intuition for this is that if the right subproblem selects $R_{\text{mid,right}}$ then in our given problem we must also select all rectangles in W_{left} and pay $\sum_{R \in W_{\text{left}}} c(R)$ for them. We assign all rectangles in $W_{\text{right}} \cup \{R_{\text{mid,right}}\}$ to $\mathcal{R}'_{\text{right}}$. If there is no rectangle $R_{\text{mid}} \in W$ with $\text{left}(R_{\text{mid}}) < \text{mid}(A) < \text{right}(R_{\text{mid}})$ then instead we increase the cost of the leftmost rectangle $R_{\text{leftmost}} \in W_{\text{right}}$ by $\sum_{R \in W_{\text{left}}} c(R)$, i.e., we redefine $c(R_{\text{leftmost}}) := c(R_{\text{leftmost}}) + \sum_{R \in W_{\text{left}}} c(R)$. Finally, we assign all rectangles in W_{left} to $\mathcal{R}'_{\text{left}}$ and all rectangles in W_{right} to $\mathcal{R}'_{\text{right}}$.

Right-sticking-in and spanning rows. Consider now the right-sticking-in rows $\mathcal{W}_{\text{right}}$ and let $W \in \mathcal{W}_{\text{right}}$. A simple case arises if each rectangle $R \in W \setminus \text{APX}_{\text{mid}}$ satisfies that $\text{mid}(A) \leq \text{left}(R)$. In particular, this happens for rows $W \in \mathcal{W}_{\text{right}}^{c',p'}$ below $W_{\text{right,filled}}^{c',p'}$ for some pair (c', p') for which $W_{\text{right,filled}}^{c',p'}$ is defined. In this case, we assign each rectangle in $W \setminus \text{APX}_{\text{mid}}$ to $\mathcal{R}'_{\text{right}}$. Assume now that there is a rectangle $R \in W \setminus \text{APX}_{\text{mid}}$ intersecting A_{left}. Then, we assign each rectangle in W to $\mathcal{R}'_{\text{left}}$, i.e., to the rectangles for the *left* subproblem. In particular, this may include rectangles that intersect A_{right} or that are even contained in A_{right}. However, such rectangles might be needed to satisfy the demands of rays intersecting with A_{right} which we will assign to the *right* subproblem. Therefore, when we define the rays for the left subproblem, those will include certain additional *artificial rays* $\mathcal{L}^+_{\text{right}}$ intersecting with A_{right} which will ensure that the left subproblem selects sufficiently many rectangles intersecting A_{right} and, therefore, help covering the demand of rays contained in A_{right}. We treat the spanning rows in exactly the same way as the right-sticking-in rows.

Left-sticking-in rows. Finally, we consider the left-sticking-in rows $\mathcal{W}_{\text{left}}$. Consider a row $W \in \mathcal{W}_{\text{left}}$. If each rectangle $R \in W \setminus \text{APX}_{\text{mid}}$ is contained in the interior of A then, intuitively, we already selected some rectangles in W and thus $W \setminus \text{APX}_{\text{mid}}$ behaves like a centered row. Therefore, in this case we treat $W \setminus \text{APX}_{\text{mid}}$ in exactly the same way as we treated the centered rows above. Assume now that there is a rectangle $R \in W \setminus \text{APX}_{\text{mid}}$ with $\text{left}(R) \leq \text{left}(A)$. If for each rectangle $R' \in W \setminus \text{APX}_{\text{mid}}$ we have that $R' \cap A_{\text{right}} = \emptyset$ (i.e., $\text{right}(R') \leq \text{mid}(A)$) then we assign each rectangle in $W \setminus \text{APX}_{\text{mid}}$ to $\mathcal{R}'_{\text{left}}$, i.e., to the left subproblem. On the other hand, if there is a rectangle $R' \in W \setminus \text{APX}_{\text{mid}}$ with $R' \cap A_{\text{right}} \neq \emptyset$ then we assign *all* rectangles in $W \setminus \text{APX}_{\text{mid}}$ to $\mathcal{R}'_{\text{right}}$, i.e., to the right subproblem. Similarly as for the right-sticking-in and the spanning rows, we will define artificial rays in $\mathcal{L}^+_{\text{left}}$ for the right subproblem to ensure that from such rows, the right subproblem selects sufficiently many rectangles to cover enough demand from rays in $\mathcal{L}'$ that intersect with A_{left}.

Reference solutions. For the left and right subproblem, we define reference solutions S^+_{left} and S^+_{right}. Intuitively, to define S^+_{left} we restrict S^+ to the rectangles contained in the left subproblem and we define S^+_{right} similarly. Moreover, whenever we cut a rectangle from S into two pieces, we assign the left piece to S^+_{left} and the right piece to S^+_{right}. Formally, we define $S^+_{\text{left}} := \{R \in \mathcal{R}_{\text{left}} : \exists R' \in S^+ \text{ with } R \subseteq R'\}$ and $S^+_{\text{right}} := \{R \in \mathcal{R}_{\text{right}} : \exists R' \in S^+ \text{ with } R \subseteq R'\}$.

Rays and artificial rays for subproblems. It remains to define the sets of rays for the left and right subproblem, respectively. As mentioned above, for a ray $L \in \mathcal{L}'$ with $L \cap A_{\text{left}} \neq \emptyset$ we would like that its demand is partially satisfied by rectangles from $\mathcal{R}'_{\text{left}}$, i.e., selected by the left subproblem, and partially by rectangles from $\mathcal{R}'_{\text{right}}$, i.e., selected by the right subproblem. Therefore, for each ray $L \in \mathcal{L}'$ with $L \cap A_{\text{left}} \neq \emptyset$ we intuitively reduce its demand by a certain value; formally, we introduce a ray in a set $\mathcal{L}'_{\text{left}}$ for the left subproblem corresponding to L with reduced demand. On the other hand, we introduce artificial rays in a set $\mathcal{L}^+_{\text{left}}$ for the right subproblem to compensate for this reduction. We perform a symmetric operation for the rays in the right subproblem.

Formally, the rays $\mathcal{L}'_{\text{left}}$ and $\mathcal{L}'_{\text{right}}$ with their reduced demands and the artificial rays $\mathcal{L}^+_{\text{left}}$ and $\mathcal{L}^+_{\text{right}}$ are defined by a function $f : A \to \{0, \dots, \sum_R p(R)\}$ which we will guess; this function is a step-function with only polylogarithmically many steps. Its steps are defined by a partition $\mathcal{Q} = \{Q_1, ..., Q_k\}$ of A such that each $Q \in \mathcal{Q}$ is an axis-parallel rectangle of the form $Q = [x^L_Q, x^R_Q) \times [y^B_Q, y^T_Q)$ for suitable values $x^L_Q, x^R_Q, y^B_Q \in \mathbb{N}$ and $y^T_Q \in \mathbb{N} \cup \{\infty\}$ and either $Q \subseteq A_{\text{left}}$ or $Q \subseteq A_{\text{right}}$. Also, for any two points $(t, s), (t', s') \in Q$ we have that $f(t, s) = f(t', s')$. For each ray $L(s, t) \in \mathcal{L}'$ with $L \subseteq A_{\text{left}}$ we add a ray $L'(s, t)$ to $\mathcal{L}'_{\text{left}}$ with a demand of $d(L'(s, t)) := d(L(s, t)) - f(t, s) - p(\{R \in \text{APX}^{\text{mid}} : R \cap L(s, t) \neq \emptyset\})$. Symmetrically, for each ray $L(s, t) \in \mathcal{L}'$ with $L \subseteq A_{\text{right}}$ we add a ray $L'(s, t)$ to $\mathcal{L}'_{\text{right}}$ with a demand of $d(L'(s, t)) := d(L(s, t)) - f(t, s) - p(\{R \in \text{APX}^{\text{mid}} : R \cap L(s, t) \neq \emptyset\})$. Additionally, we define a polylogarithmic number of artificial rays $\mathcal{L}^+_{\text{left}}$ and $\mathcal{L}^+_{\text{right}}$. For each "step" $Q \in \mathcal{Q}$ of f we introduce an artificial ray L_Q that starts in the point $(x^R_Q - 1, y^B_Q)$ (i.e., at the bottom-right integer point of Q) and is oriented vertically downwards. We define its demand such that $d(L_Q) := f(t, s)$ for each $(t, s) \in Q$. For each $Q \in \mathcal{Q}$ with $Q \subseteq A_{\text{left}}$ we denote by $\mathcal{L}^+_{\text{left}}$ the resulting set of rays, i.e., $\mathcal{L}^+_{\text{left}} = \{L_Q : Q \subseteq A_{\text{left}}\}$; similarly, we define $\mathcal{L}^+_{\text{right}} = \{L_Q : Q \subseteq A_{\text{right}}\}$. Note that all rays in $\mathcal{L}'_{\text{left}}, \mathcal{L}'_{\text{right}}, \mathcal{L}^+_{\text{left}}$, and $\mathcal{L}^+_{\text{right}}$ are uniquely defined from f. Therefore, we say that they are *induced* by f.

It remains to consider the rays in L'_{out}. We introduce sets of rays $\mathcal{L}'_{\text{left,out}}, \mathcal{L}'_{\text{right,out}}$ (independent of f) for the left and right subproblems, respectively. Consider a ray $L(s, t) \in \mathcal{L}'_{\text{out}}$. We guess the value $p(\{R \in S^+_{\text{left}} : R \cap L(s, t) \neq \emptyset\})$ by which the reference solution for the left subproblem covers $L(s, t)$.

Then we add a ray $L'(s,t)$ with a demand of $d(L'(s,t)) := p(\{R \in S^+_{\text{left}} : R \cap L(s,t) \neq \emptyset\})$ to $\mathcal{L}'_{\text{left,out}}$ and a ray $L''(s,t)$ with $d(L''(s,t)) := \max\{0, d(L(s,t)) - d(L'(s,t)) - p(\{R \in \text{APX}^{\text{mid}} : R \cap L(s,t) \neq \emptyset\})\}$ to $\mathcal{L}'_{\text{left,out}}$.

In the next lemma, we show that there exists a function f and corresponding induced rays which admit certain properties. Those will allow us to partition the remaining problem into the left and the right subproblem.

Lemma 16. *There exists a step-function $f : A \to \{0, \dots, \sum_R p(R)\}$ with only $(K \cdot M \log(T+p_{\max})/\varepsilon)^{O(1)}$ steps with the following properties. Let $\mathcal{L}'_{\text{left}}, \mathcal{L}'_{\text{right}}, \mathcal{L}^+_{\text{left}}$, and $\mathcal{L}^+_{\text{right}}$ be the rays induced by f. It holds that*

- $|\mathcal{L}^+_{\text{left}} \cup \mathcal{L}^+_{\text{right}}| \leq (K \cdot M \cdot \log(T \cdot p_{\max})/\varepsilon)^{O(1)}$
- *each ray $L \in \mathcal{L}^+_{\text{left}}$ is contained in A_{left} and each ray $L \in \mathcal{L}^+_{\text{right}}$ is contained in A_{right},*
- *the solution S^+_{left} is a feasible solution to the (left) subproblem $(A_{\text{left}}, \mathcal{L}'_{\text{left}} \cup \mathcal{L}^+_{\text{right}} \cup \mathcal{L}'_{\text{left,out}}, \mathcal{R}_{\text{left}})$,*
- *the solution S^+_{right} is a feasible solution to the (right) subproblem $(A_{\text{right}}, \mathcal{L}'_{\text{right}} \cup \mathcal{L}^+_{\text{left}} \cup \mathcal{L}'_{\text{right,out}}, \mathcal{R}_{\text{right}})$.*

We will prove this lemma to Section 4.4. Algorithmically, we guess f which we can do in time $2^{(K \cdot M \cdot \log(n \cdot T \cdot p_{\max})/\varepsilon)^{O(1)}}$ since we have that $|\mathcal{Q}| \leq (K \cdot M \cdot \log(T \cdot p_{\max})/\varepsilon)^{O(1)}$ and for the value of f corresponding to each $Q \in \mathcal{Q}$ (i.e., the value $f(s,t)$ for each $(s,t) \in Q$) there are only $O(n \cdot p_{\max})$ options. We recurse on the left and right subproblems $(A_{\text{left}}, \mathcal{L}_{\text{left}} \cup \mathcal{L}^+_{\text{right}} \cup \mathcal{L}'_{\text{left,out}}, \mathcal{R}_{\text{left}})$ and $(A_{\text{right}}, \mathcal{L}_{\text{right}} \cup \mathcal{L}^+_{\text{left}} \cup \mathcal{L}'_{\text{right,out}}, \mathcal{R}_{\text{right}})$. Let APX_{left} and $\text{APX}_{\text{right}}$ denote the obtained solutions for them. Intuitively, we output $\text{APX}_{\text{mid}} \cup \text{APX}_{\text{left}} \cup \text{APX}_{\text{right}}$. Formally, we need to include also all rectangles on the left of the rectangles in these sets. Therefore, our output APX is the set of all rectangles $R = [\text{left}(R), \text{right}(R)) \times [j, j+1) \in \mathcal{R}$ for which there exists a rectangle $R' = [\text{left}(R'), \text{right}(R')) \times [j, j+1) \in \text{APX}_{\text{mid}} \cup \text{APX}_{\text{left}} \cup \text{APX}_{\text{right}}$ with $\text{left}(R) < \text{right}(R')$.

4.3 Analysis

For proving Lemma 4, we need to show that our computed solution is feasible, prove that it has the claimed approximation ratio, and bound the running time of our algorithm. Let APX^{root} denote the solution obtained for the root subproblem, i.e., where $A = [0,T) \times [0,\infty)$, $\mathcal{R}' = \mathcal{R}$ and $\mathcal{L}' = \mathcal{L}$. First, we prove feasibility.

Lemma 17. *The set APX^{root} is a feasible solution for the given instance of RCP.*

Proof. We show by induction, that the set APX computed as a solution for a given subproblem $(A, \mathcal{L}', \mathcal{R}')$ is a feasible solution to the subproblem $(A, \mathcal{L}', \mathcal{R}')$. Consider a subproblem $(A, \mathcal{L}', \mathcal{R}')$. If $\text{left}(A) - \text{right}(A) = 1$ then the claim follows from Lemma 13. So suppose this is not the case. We have to show that APX contains a prefix from the rectangles in each row and that it covers the rays $\mathcal{L}'$. Recall that given $\text{APX}_{\text{left}}, \text{APX}_{\text{right}}$ and APX_{mid}, for each row $W \in \mathcal{W}$ we select a rectangle $R = [\text{left}(R), \text{right}(R)) \times [j, j+1)$ if there exists a rectangle $R' = [\text{left}(R'), \text{right}(R')) \times [j, j+1) \in \text{APX}_{\text{left}} \cup \text{APX}_{\text{right}} \cup \text{APX}_{\text{mid}}$ with $\text{right}(R') > \text{left}(R)$. This directly implies that APX contains a prefix in each row. So it remains to show that each ray is covered by APX.

Consider a ray $L(s,t) \in \mathcal{L}'$. As a first step, we show that APX covers at least as much as APX_{left}, $\text{APX}_{\text{right}}$ and APX_{mid} together. Let $W \in \mathcal{W}$ be a row and let $R = [\text{left}(R), \text{right}(R)) \times [j, j+1) \in W$ be a rectangle in this row. Furthermore let $W' := \{R' = [\text{left}(R'), \text{right}(R')) \times [j', j'+1) \in \mathcal{R}'_{\text{left}} \cup \mathcal{R}'_{\text{right}} \cup \text{APX}_{\text{mid}} : j' = j\}$ denote the rectangles that, intuitively, are in the row W, but in one of the

subproblems. Recall that for some rows, we might split a rectangle into two parts. By construction, for every t there is at most one rectangle $R \in W'$ with $\text{left}(R) \leq t < \text{right}(R)$. Thus $L(s,t)$ can intersect with at most one rectangle from W'. So it also intersects with at most one rectangle from $(\text{APX}_{\text{left}} \cup \text{APX}_{\text{right}} \cup \text{APX}_{\text{mid}}) \cap W'$. Note that all rectangles $R \in W'$ have the same value $p(R)$. Suppose that there is a rectangle $R \in (\text{APX}_{\text{left}} \cup \text{APX}_{\text{right}} \cup \text{APX}_{\text{mid}}) \cap W'$ with $R \cap L(s,t) \neq \emptyset$. Then there is also a rectangle $R' \in W$ with $R' \cap L(s,t) \neq \emptyset$. Furthermore we have $\text{left}(R') < t \leq \text{right}(R)$. This implies $R' \in \text{APX}$ as we add a rectangle $R' \in W$ to APX if there is a rectangle $R \in W'$ with $\text{left}(R') < \text{right}(R)$. So we obtain $p(\{R \in \text{APX} \cap W : R \cap L(s,t) \neq \emptyset\}) \geq p(\{R \in (\text{APX}_{\text{left}} \cup \text{APX}_{\text{right}} \cup \text{APX}_{\text{mid}}) \cap W' : R \cap L(s,t) \neq \emptyset\})$ and therefore

$$\begin{aligned} &p(\{R \in \text{APX} : R \cap L(s,t) \neq \emptyset\}) \geq p(\{R \in \text{APX}_{\text{left}} : R \cap L(s,t) \neq \emptyset\}) \\ &+ p(\{R \in \text{APX}_{\text{right}} : R \cap L(s,t) \neq \emptyset\}) + p(\{R \in \text{APX}_{\text{mid}} : R \cap L(s,t) \neq \emptyset\}) \end{aligned}$$

Now we prove that APX covers $L(s,t)$. First suppose that $L(s,t) \in \mathcal{L}'_{\text{out}}$. Then there is a ray $L'(s,t)$ with demand $d(L'(s,t))$ in the left subproblem and a ray with $L''(s,t)$ with demand $d(L''(s,t))$ in the right subproblem. Note that APX_{left} and $\text{APX}_{\text{right}}$ cover the demands of these rays in the respective subproblems, as they are feasible solutions for them. By definition of $d(L''(s,t))$ we have $d(L(s,t)) \leq d(L'(s,t)) + d(L''(s,t)) + p(\{R \in \text{APX}_{\text{mid}} : R \cap L(s,t) \neq \emptyset\})$. So we obtain the following:

$$\begin{aligned} p(\{R \in \text{APX} : R \cap L(s,t) \neq \emptyset\}) &\geq p(\{R \in \text{APX}_{\text{left}} : R \cap L(s,t) \neq \emptyset\}) + p(\{R \in \text{APX}_{\text{right}} : R \cap L(s,t) \neq \emptyset\}) \\ &\quad + p(\{R \in \text{APX}_{\text{mid}} : R \cap L(s,t) \neq \emptyset\}) \\ &\geq d(L'(s,t)) + d(L''(s,t)) + p(\{R \in \text{APX}_{\text{mid}} : R \cap L(s,t) \neq \emptyset\}) \\ &\geq d(L(s,t)) \end{aligned}$$

Hence, $L(s,t)$ is covered by APX. The same argument yields that every ray in $\mathcal{L}'_{\text{out}}$ is covered by APX.

Now suppose that $L(s,t)$ intersects A_{left}. Then there is a step $Q \in \mathcal{Q}$ of the function f with $(t,s) \in Q$ and thus $Q \subseteq A_{\text{left}}$. Let $\bar{L}(\bar{s},\bar{t}) \in \mathcal{L}^+_{\text{left}}$ be the ray at the bottom-right integer point of Q. Then in the left subproblem, there is a ray $L'(s,t)$ with demand $d(L'(s,t)) = \max\{0, d(L(s,t)) - f(t,s) - p(\{R \in \text{APX}_{\text{mid}} : R \cap L(s,t) \neq \emptyset\})\}$ and in the right subproblem, there is a the demand ray $\bar{L}(\bar{s},\bar{t})$ with demand $d(\bar{L}(\bar{s},\bar{t})) = f(t,s)$. As before, let W be a row, let $R = [\text{left}(R), \text{right}(R)) \times [j, j+1) \in W$ and let $W' := \{R' = [\text{left}(R'), \text{right}(R')) \times [j', j'+1) \in \mathcal{R}'_{\text{left}} \cup \mathcal{R}'_{\text{right}} \cup \text{APX}_{\text{mid}} : j' = j\}$ denote the rectangles that, intuitively, are in the row W, but in one of the subproblems. Suppose that there exists a rectangle $R \in W' \cap \text{APX}_{\text{right}}$ with $R \cap \bar{L}(\bar{s},\bar{t}) \neq \emptyset$. Note that there is at most one such rectangle in each row. As $t < \text{mid}(A)$ and $Q \subseteq A_{\text{left}}$ by construction, we also have $\bar{t} < \text{mid}(A)$ and thus $\text{left}(R) < \text{mid}(A)$. By construction of the rectangles $\mathcal{R}_{\text{right}}$, this is only possible if there also exists a rectangle $R' \in W'$ with $\text{left}(R') \leq \text{left}(A)$. So there exists a rectangle $R \in W'$ with $\text{right}(R) > \bar{t} \geq t$ and a rectangle $R' \in W'$ with $\text{left}(R') \leq \text{left}(A) \leq t$. As the rectangles in a row are consecutive there also exists a rectangle $R'' \in W'$ with $\text{left}(R'') \leq t < \text{right}(R'')$. As the algorithm selects a prefix in each row and $R \in \text{APX}_{\text{right}}$, we also have $R'' \in \text{APX}_{\text{right}}$. This implies $R'' \cap L(s,t) \neq \emptyset$ as $s \geq \bar{s}$. So $p(\{R \in \text{APX}_{\text{right}} : R \cap L(s,t) \neq \emptyset\}) \geq p(\{R \in \text{APX}_{\text{right}} : R \cap \bar{L}(\bar{s},\bar{t}) \neq \emptyset\}) \geq d(\bar{L}(\bar{s},\bar{t})) = f(t,s)$. This yields the desired result:

$$\begin{aligned} p(\{R \in \text{APX} : R \cap L(s,t) \neq \emptyset\}) &\geq p(\{R \in \text{APX}_{\text{left}} : R \cap L(s,t) \neq \emptyset\}) + p(\{R \in \text{APX}_{\text{right}} : R \cap L(s,t) \neq \emptyset\}) \\ &\quad + p(\{R \in \text{APX}_{\text{mid}} : R \cap L(s,t) \neq \emptyset\}) \\ &\geq d(L'(s,t)) + f(t,s) + p(\{R \in \text{APX}_{\text{mid}} : R \cap L(s,t) \neq \emptyset\}) \\ &\geq d(L(s,t)) \end{aligned}$$

The proof for the case that $L(s,t)$ intersects A_{right} is the same as the proof for the case that $L(s,t)$ intersects A_{left} when interchanging left and right. This shows that every ray is covered and thus APX is a feasible solution for $(A, \mathcal{L}', \mathcal{R}')$. This completes the induction.

Thus the set APX^{root} is a feasible solution to the root subproblem. And as the root subproblem is equivalent to the RCP instance, the set APX^{root} is also feasible for the RCP instance. □

As a next step, we bound our approximation ratio.

Lemma 18. *We have $c(\text{APX}^{\text{root}}) \leq (2 + O(\varepsilon))c(S)$ for any feasible solution S.*

We will prove this lemma in Section 4.5. As a final step we bound our running time.

Lemma 19. *The running time of the algorithm is bounded by $2^{(K \cdot M \cdot \log(n \cdot T \cdot P)/\varepsilon)^{O(1)}}$.*

Proof. As a first step, we show that the size of $\mathcal{L}'_{\text{out}}$ is at most $(K \cdot M \cdot \log(T \cdot p_{\max})/\varepsilon)^{O(1)}$. Let C_1 be a constant such that the function f from Lemma 16 has at most $(K \cdot M \log(T + p_{\max})/\varepsilon)^{C_1}$ steps. We show by induction starting from the root, that for a subproblem with area $A = [\text{left}(A), \text{right}(A)) \times [0, \infty)$, we have $|\mathcal{L}'_{\text{out}}| \leq (\log T - \log(\text{right}(A) - \text{left}(A))) \cdot (K \cdot M \cdot \log(T + p_{\max})/\varepsilon)^{C_1}$. For the root we have $\text{right}(A) - \text{left}(A) = T$ and there are 0 rays outside of A, so this is correct. So suppose the the hypothesis holds for a call for A and we show that it also holds for the right and left subproblem. The rays in $\mathcal{L}'_{\text{out}}$ are passed on to both subproblems (and do not intersect with A_{left} and A_{right}). These are at most $(\log T - \log(\text{right}(A) - \text{left}(A))) \cdot (K \cdot M \cdot \log(T + p_{\max})/\varepsilon)^{C_1}$. The other rays not intersecting A_{left} in the left subproblem are the rays $\mathcal{L}^+_{\text{right}}$. By Lemma 16, these are at most $(K \cdot M \cdot \log(T + p_{\max})/\varepsilon)^{C_1}$. This in total yields $(\log T - \log(\text{left}(A) - \text{right}(A)) + 1) \cdot (K \cdot M \cdot \log(T + p_{\max})/\varepsilon)^{C_1} = (\log T - \log((\text{left}(A) - \text{right}(A))/2)) \cdot (K \cdot M \cdot \log(T + p_{\max})/\varepsilon)^{C_1}$ not intersecting A_{left} in the left subproblem. The same argument yields the same bound for the right subproblem right subproblem. So we always have that the size of $\mathcal{L}'_{\text{out}}$ is at most $\log T \cdot (K \cdot M \log(T \cdot p_{\max})/\varepsilon)^{C_1}$.

Using this, we can now prove that the running time of the algorithm is bounded by $2^{(K \cdot M \cdot \log(n \cdot T \cdot P)/\varepsilon)^{O(1)}}$. The recursion depth of our algorithm is $O(\log T)$. At each recursion step, we need to guess the numbers for each pair c', p') and $\text{set} \in \{\text{left}, \text{span}, \text{right}\}$ we guess whether $n^{c',p'}_{\text{set}} < \frac{2K \cdot \log T}{\varepsilon}$ and up to $\frac{2K \cdot \log T}{\varepsilon}$ rows. For each row, there are at most n options. As there are at most M values for c' and at most $O(\log p_{\max}/\varepsilon)$ possible values for p', there are at most $O(M \log p_{\max}/\varepsilon)$ pairs (c', p'). So this step takes at most $2^{O((M \log T \log p_{\max}/\varepsilon)^2)}$.

Furthermore, we need to guess the function f. The function has at most $(K \cdot M \cdot \log(T \cdot p_{\max})/\varepsilon)^{O(1)}$ steps. For each step $Q \in \mathcal{Q}$, we need to guess the function value in Q, which is bounded by $n \cdot p_{\max}$, and the boundaries of Q, for which there are at most n options for the top and bottom one and T options for the left and right one. So altogether, the guessing of f can be done in $2^{(K \cdot M \cdot \log(n \cdot T \cdot p_{\max})/\varepsilon)^{O(1)}}$.

The last step is to guess the demand for the rays in $\mathcal{L}'_{\text{out}}$. There are at most $(K \cdot M \cdot \log(T \cdot p_{\max})/\varepsilon)^{O(1)}$ such rays as shown above. And for each such ray, the guessed demand can be bounded by $n \cdot p_{\max}$, so this step takes $2^{(K \cdot M \cdot \log(n \cdot T \cdot p_{\max})/\varepsilon)^{O(1)}}$ Altogether, at each recursion step there are at most $2^{(K \cdot M \log(n+T+p_{\max})/\varepsilon)^{O(1)}}$ recursive calls.

For fixed guesses, the computation of APX can be done in time $O(n)$. The base case can be solved in time $(n \cdot p_{\max})^{O(|\mathcal{L}'_{\text{out}}|)} \leq 2^{(K \cdot M \cdot \log(n \cdot T \cdot p_{\max})/\varepsilon)^{O(1)}}$. Altogether, this yields a running time of $2^{(K \cdot M \cdot \log(n \cdot T \cdot p_{\max})/\varepsilon)^{O(1)}}$. □

Altogether, we can now prove Lemma 4.

Proof of Lemma 4. By Lemma 17 the computed solution $\mathrm{APX}^{\mathrm{root}}$ is feasible. By Lemma 18 we have $c(\mathrm{APX}) \leq (2 + O(\varepsilon))c(S)$where S denotes the optimal solution to the RCP instance. By Lemma 19, the running time is bounded by $2^{(K \cdot M \cdot \log(n \cdot T \cdot P)/\varepsilon)^{O(1)}}$. So we can apply Lemma 12, which yields the claimed result by rescaling ε. □

4.4 Proof of Lemma 16

We construct the step function f separately for the left and right subproblem. First, we introduce some notation. Let $\mathcal{G}$ be the set of all pairs $(c', p') \in \mathbb{N}_0^2$, for which there exists $\mathrm{set} \in \{\mathrm{left}, \mathrm{right}, \mathrm{span}\}$ such that $\mathcal{W}_{\mathrm{set}}^{c',p'} \neq \emptyset$. For each $\mathrm{set} \in \{\mathrm{left}, \mathrm{right}, \mathrm{span}\}$ and a set of rows $\mathcal{W}_{\mathrm{set}}^{c',p'}$ for which $W_{\mathrm{set}}^{c',p'}$ was defined let $\mathcal{W}_{\mathrm{set,top}}^{c',p'}$ denote all rows in $\mathcal{W}_{\mathrm{set}}^{c',p'}$ above $W_{\mathrm{set,filled}}^{c',p'}$. Also let $\mathcal{H}_{\mathrm{set}}$ denote all pairs (c', p') for which $\mathcal{W}_{\mathrm{set,top}}^{c',p'} \neq \emptyset$. For each $\mathrm{set} \in \{\mathrm{left}, \mathrm{right}, \mathrm{span}\}$, we apply the following to lemma to $\bar{W} := \bigcup_{c',p' \in \mathcal{H}_{\mathrm{set}}} \mathcal{W}^{c',p'}$, to the rectangles $\bar{S} := S^+ \cap \bar{W}$ and the area $\bar{A} = A_{\mathrm{left}}$ if $\mathrm{set} = \mathrm{left}$ and the area $\bar{A} = A_{\mathrm{right}}$ if $\mathrm{set} \in \{\mathrm{span}, \mathrm{right}\}$.

Lemma 20. *Let $\bar{A} = [\bar{a}, \bar{b}) \times [0, \infty)$ be an area and let $\bar{\mathcal{W}} \subseteq \mathcal{W}$ be rows spanning $\bar{A}$, i.e. for each $W \in \bar{\mathcal{W}}$ there exist rectangles $R, R' \in W$ with $\mathrm{left}(R) \leq \bar{a}$ and $\mathrm{right}(R') \geq \bar{b}$. Let $\bar{\mathcal{W}} = \bigcup_{c',p'} \bar{\mathcal{W}}^{c',p'}$ be a partition, where a row $W \in \bar{\mathcal{W}}$ if and only if the leftmost rectangle $R \in W$ has a cost of $c(R) = c'$ and satisfies $(1+\varepsilon)^{p'} \leq p(R) \leq (1+\varepsilon)^{p'+1}$. In addition, let $\bar{S} \subseteq \bigcup_{W \in \bar{\mathcal{W}}} W$ be a set of rectangles that contains a prefix of the rectangles in each row $W \in \bar{\mathcal{W}}$. For each (c', p') let $\bar{n}^{c',p'} := |\{W \in \bar{\mathcal{W}}^{c',p'} : W \cap \bar{S} \neq \emptyset\}|$. There exists a step function $\bar{f} : A \to \{0, \ldots, \sum_R p(R)\}$ with $O((K|\mathcal{G}| \log T/\varepsilon)^2)$ steps such that for each $s, t \in \bar{A}$ we have*

$$\bar{f}(t,s) \leq \sum_{R \in \bar{S} : R \cap L(s,t) \neq \emptyset} p(R) \leq \bar{f}(t,s) + \sum_{(c',p') : \exists R \in \bigcup_{W \in \bar{\mathcal{W}}^{c',p'}} W \text{ with } R \cap L(s,t) \neq \emptyset} (1+\varepsilon)^{p'} \left\lfloor \frac{\varepsilon}{2 \cdot K \log T} \bar{n}^{c',p'} \right\rfloor. \tag{3}$$

Proof. Let $g(t,s) := p(\{R \in \bar{S} : R \cap L(s,t) \neq \emptyset\})$ be the total amount covered on a ray $L(s,t)$ for each $(t,s) \in \bar{A}$. Note that as we select a prefix in each row and all rows are spanning $\bar{A}$, the function $g(t,s)$ is non-increasing in t (for each fixed s). And as the rays are downward oriented, the function is non-decreasing in s (for fixed t). Now we need to show that we can approximate g by a function $\bar{f}$ with few steps. Let $X = \frac{8 \cdot K \cdot \log T}{\varepsilon}$. For each row $W \in \mathcal{W}$ and $R = [\mathrm{left}(R), \mathrm{right}(R)) \times [j, j+1) \in W$ let $\mathrm{proj_y}(W) := j$ be the y-coordinate of the rectangles in W. Consider a pair (c', p'). Intuitively, we first split A at certain horizontal lines, such that between two consecutive lines, the value of g does not change a lot because there are not a lot of rows between two consecutive lines. Formally we show that there exists a set $\mathcal{V}^{c',p'} = \{V_0, \ldots V_k\} \subseteq \bar{\mathcal{W}}^{c',p'}$ with $k \leq X$ such that V_0 is the bottom row in $\bar{\mathcal{W}}^{c',p'}$, V_k is the top row in $\bar{\mathcal{W}}^{c',p'}$ and for each $k' < k$ we have $|\{W \in \bar{\mathcal{W}}^{c',p'} : \mathrm{proj_y}(V_{k'}) < \mathrm{proj_y}(W) < \mathrm{proj_y}(V_{k'+1})\}| \leq \frac{\bar{n}^{c',p'}}{X}$. If $|\bar{\mathcal{W}}^{c,p}| \leq X$, we chose $\mathcal{V}^{c,p} = \bar{\mathcal{W}}^{c,p}$. Otherwise choose V_0 V_0 as the bottom row in $\bar{\mathcal{W}}^{c',p'}$ and then recursively choose $V_{k'+1}$ with maximal $\mathrm{proj_y}(V_{k'+1})$ such that $|\{W \in \bar{\mathcal{W}}^{c',p'} : \mathrm{proj_y}(V_{k'}) < \mathrm{proj_y}(W) < \mathrm{proj_y}(V_{k'+1})\}| \leq \frac{\bar{n}^{c',p'}}{X}$. As we chose $V_{k'+1}$ with maximal $\mathrm{proj_y}(V_{k'+1})$, when we add one more row to this set, we violate the inequality, i.e., we have $|\{W \in \bar{\mathcal{W}}^{c',p'} : \mathrm{proj_y}(V_{k'}) < \mathrm{proj_y}(W) \leq \mathrm{proj_y}(V_{k'+1})\}| \geq \frac{\bar{n}^{c',p'}}{X}$ for all $k' < k-1$, which shows $k \leq X$. To simplify notation let $\mathrm{proj_y}(\infty) := \infty$. Let $\mathcal{V} := \bigcup_{c',p'} \mathcal{V}^{c',p'} \cup \{\infty\} = \{W_1, \ldots, W_{\ell'}\}$ be such that $\mathrm{proj_y}(W_\ell) < \mathrm{proj_y}(W_{\ell+1})$ for all $\ell < \ell'$. The y-coordinates of the steps of f are always $y_Q^B = \mathrm{proj_y}(W_\ell)$ and $y_Q^T = \mathrm{proj_y}(W_{\ell+1})$ for some $\ell < \ell'$.

Let $W_\ell \in \mathcal{V}$ with $\ell < \ell'$. We will define f such that each step Q of f fulfills $y_Q^B = \mathrm{proj_y}(W_\ell)$ and $y_Q^T = \mathrm{proj_y}(W_{\ell+1})$. Towards this, let $s' := \mathrm{proj_y}(W_\ell)$ and let $\mathcal{H} := \{(c', p') : \exists W \in \bar{\mathcal{W}}^{c',p'} \text{ with } \mathrm{proj_y}(W) \leq s'\}$ denote all pairs for which there exists a row below W_ℓ. We show that there exists a set $\{(t'_0, s'), \dots, (t'_k, s')\}$ with $k \leq X \cdot |\mathcal{G}|$ such that $t'_0 = \mathrm{left}(\bar{A})$, $t'_k = \mathrm{right}(\bar{A})$ and for each $k' < k$ we have $g(t'_{k'}, s') - g(t'_{k'+1} - 1, s') \leq \sum_{(c',p')\in\mathcal{H}} (1+\varepsilon)^{p'+1} \lfloor \frac{\bar{n}^{c',p'}}{X} \rfloor$. If $\mathrm{right}(\bar{A}) - \mathrm{left}(\bar{A}) \leq X$, we can just chose $t'_k = \mathrm{left}(\bar{A}) + k$. Otherwise choose t'_0 as required and then recursively choose $t_{k'+1}$ maximal such that $g(t'_{k'}, s') - g(t'_{k'+1} - 1, s') \leq \sum_{(c',p')\in\mathcal{H}} (1+\varepsilon)^{p'+1} \lfloor \frac{\bar{n}^{c',p'}}{X} \rfloor$. By doing this, for each $k' \leq k-2$ we have $g(t'_{k'}, s') - g(t'_{k'+1}, s') \geq (1+\varepsilon)^{p'+1} \frac{\bar{n}^{c',p'}}{X}$ for some group $(c', p') \in \mathcal{H}$. As $g(t'_0, s') \leq \sum_{(c',p')\in\mathcal{H}} (1+\varepsilon)^{p'+1} \bar{n}^{c',p'}$, this shows

$$k \leq \sum_{(c',p')\in\mathcal{H}} \frac{(1+\varepsilon)^{p'+1} \bar{n}^{c',p'}}{(1+\varepsilon)^{p'+1} \cdot \bar{n}^{c',p'}/X} = X \cdot |\mathcal{H}| \leq X \cdot |\mathcal{G}|.$$

We have a step $Q = [t'_{k'}, t'_{k'+1}) \times [s', \mathrm{proj_y}(W_{\ell+1}))$, i.e. for each (t, s) with $s' \leq s < \mathrm{proj_y}(W_{\ell+1})$ and $t'_{k'} \leq t < t'_{k'+1}$ let $\bar{f}(t, s) := g(t'_{k'+1} - 1, s')$ be the value of g at the bottom-right integer point of Q.

It remains to show that $\bar{f}$ fulfills the requirements of the lemma. First note that the number of steps is bounded by $(X \cdot |\mathcal{G}| + 1)|V| \leq (X+1)^2 |\mathcal{G}|^2 \leq O((C|\mathcal{G}| \log T/\varepsilon)^2)$.

Now fix a step $Q = [x_Q^L, x_Q^R) \times [y_Q^B, y_Q^T)$ and let $(t, s) \in Q$. Let $\ell < \ell'$ such that $y_Q^B = \mathrm{proj_y}(W_\ell)$ and $y_Q^T = \mathrm{proj_y}(W_{\ell+1})$. As $g(t', s')$ is non-increasing in t' and non-decreasing in s', we have that $\min_{(t',s')\in Q\cap\mathbb{N}_0^2} g(t', s') = g(x_Q^T - 1, y_Q^B)$. So $g(t, s) \geq g(x_Q^R - 1, y_Q^B) = \bar{f}(t, s)$, which is the left inequality in (3). Thus, only the right inequality in (3) remains to be proven. By construction of $\mathcal{V}$, we know that the values $g(t, s)$ and $g(t, y_Q^B)$ do not differ by much, as there are not many rows between s and y_Q^B. Formally, we have $g(t, s) - g(t, y_Q^B) \leq \sum_{(c',p')\in\mathcal{H}} (1+\varepsilon)^{p'+1} \lfloor \frac{\bar{n}^{c,p}}{X} \rfloor$ as rectangles intersecting with the ray $L(s, t)$, but not the ray $L(y_Q^B, t)$ can only be in rows belonging to groups in $\mathcal{H}$, from each group $(c', p') \in \mathcal{H}$ there can be only $\lfloor \frac{\bar{n}^{c',p'}}{X} \rfloor$ rows and within a row there can be only one rectangle intersecting with the ray.

When we defined $t'_{k'}$, we ensured that $g(t, y_Q^B) - g(x_Q^R - 1, y_Q^B) \leq \sum_{(c',p')\in\mathcal{H}} (1+\varepsilon)^{p'+1} \lfloor \frac{\bar{n}^{c',p'}}{X} \rfloor$. Altogether, this implies

$$\begin{aligned} g(t, s) - \bar{f}(t, s) &= g(t, s) - g(x_Q^R - 1, y_Q^B) \\ &\leq 2 \sum_{(c',p')\in\mathcal{H}} (1+\varepsilon)^{p'+1} \lfloor \frac{\bar{n}^{c',p'}}{X} \rfloor \\ &\leq \sum_{(c',p')\in\mathcal{H}} (1+\varepsilon)^{p'} \lfloor \frac{\varepsilon \cdot \bar{n}^{c',p'}}{2 \cdot K \cdot \log T} \rfloor \end{aligned}$$

This completes the proof of the lemma. □

For each $\mathrm{set} \in \{\mathrm{left}, \mathrm{right}, \mathrm{span}\}$ let f_{set} be the function obtained from applying Lemma 20 to $\bar{W} := \bigcup_{(c',p')\in\mathcal{H}_{\mathrm{set}}} \mathcal{W}^{c',p'}$, to the rectangles $\bar{S} := S^+ \cap \bar{W}$ and the area $\bar{A} = A_{\mathrm{left}}$ if $\mathrm{set} = \mathrm{left}$ and the area $\bar{A} = A_{\mathrm{right}}$ if $\mathrm{set} \in \{\mathrm{span}, \mathrm{right}\}$. Let

$$f(t, s) = \begin{cases} f_{\mathrm{left}}(t, s) & \text{if } (t, s) \in A_{\mathrm{left}}) \\ f_{\mathrm{right}}(t, s) + f_{\mathrm{span}}(t, s) & \text{if } (t, s) \in A_{\mathrm{right}} \end{cases}$$

The function f has at most $O((K|\mathcal{G}|\log T/\varepsilon)^2)$ steps in A_{left}. Each step of f in A_{right} is the intersection of a step of f_{right} and a step of f_{span}, so f has at most $O((K|\mathcal{G}|\log T/\varepsilon)^4)$ steps in A_{right}. Therefore, we also only have $O((K|\mathcal{G}|\log T/\varepsilon)^4)$ rays in $\mathcal{L}^+_{\text{left}}$ and $\mathcal{L}^+_{\text{right}}$. For each $(c',p') \in \mathcal{G}$ we have that c' is one of at most M different values and p' is one of at most $O(\log p_{\max}/\varepsilon)$ different values, thus $|\mathcal{G}| = O(M \log p_{\max}/\varepsilon)$. So the number of steps of f (and thus the number of rays in $\mathcal{L}^+_{\text{left}} \cup \mathcal{L}^+_{\text{right}}$) is bounded by $(K \cdot M \log(T+p_{\max})/\varepsilon)^{O(1)}$ steps.

So it remains to show that S^+_{left} is a feasible solution for the left subproblem and S^+_{right} is a feasible solution for the right subproblem.

Lemma 21. *For each ray $L(s,t) \in \mathcal{L}'$ we have*

$$\begin{aligned} p(\{R \in S^+ : R \cap L(s,t) \neq \emptyset\}) &= p(\{R \in S^+_{\text{left}} : R \cap L(s,t) \neq \emptyset\}) \\ &+ p(\{R \in S^+_{\text{right}} : R \cap L(s,t) \neq \emptyset\}) \\ &+ p(\{R \in \text{APX}_{\text{mid}} : R \cap L(s,t) \neq \emptyset\}) \end{aligned}$$

Proof. Let $W \in \mathcal{W}$ be a row, let $R = [\text{left}(R), \text{right}(R)) \times [j, j+1) \in W$ be a rectangle in this row and let $W' := \{R' = [\text{left}(R'), \text{right}(R')) \times [j', j'+1) \in \mathcal{R}_{\text{left}} \cup \mathcal{R}_{\text{right}} \cup \text{APX}_{\text{mid}} : j' = j\}$ denote, intuitively, the rectangles in row W in the subproblems. Let $L(s,t) \in \mathcal{L}'$. By construction there is at most one rectangle $R \in (S^+_{\text{right}} \cup S^+_{\text{left}} \cup \text{APX}_{\text{mid}}) \cap W'$ with $\text{left}(R) \leq t < \text{right}(R)$. This implies

$$\begin{aligned} p(\{R \in S^+ \cap W : R \cap L(s,t) \neq \emptyset\}) &= p(\{R \in S^+_{\text{left}} \cap W' : R \cap L(s,t) \neq \emptyset\}) \\ &+ p(\{R \in S^+_{\text{right}} \cap W' : R \cap L(s,t) \neq \emptyset\}) \\ &+ p(\{R \in \text{APX}_{\text{mid}} \cap W : R \cap L(s,t) \neq \emptyset\}) \end{aligned}$$

And a union bound over all rows W yields the lemma. □

Now we show that S_{left} and S_{right} are feasible solutions.

Lemma 22. *The set S_{left} is a feasible solution for the left subproblem and the set S_{right} is a feasible solution for the right subproblem.*

Proof. We start with S_{left}. It follows directly from the construction of S_{left} that it contains a prefix in each row. The other necessary property for feasibility is to show that all demands are satisfied. For that, consider a ray $L(s,t) \in \mathcal{L}'_{\text{left}} \cup \mathcal{L}^+_{\text{right}} \cup \mathcal{L}'_{\text{left,out}}$.

If $L(s,t) \in \mathcal{L}'_{\text{left,out}}$, recall that $d(L(s,t)) = p(\{R \in S^+_{\text{left}} : R \cap L(s,t) \neq \emptyset\})$. So such a demand ray is covered by S^+_{left}. And if $L(t,s) \in \mathcal{L}^+_{\text{right}}$ the demand is $f(s,t)$ and as f_{left} was chosen according to Lemma 20, the solution S_{left} covers this ray.

So suppose that $L(s,t) \in \mathcal{L}'_{\text{left}}$ and let $L'(s,t) \in \mathcal{L}'$ be the ray with the same coordinates as $L(s,t)$. Then $d(L(s,t)) = d(L'(s,t)) - f(t,s) - p(\{R \in \text{APX}_{\text{mid}} : R \cap L(s,t) \neq \emptyset\})$. So the ray is covered if and only if $p(\{R \in S^+_{\text{left}} : R \cap L(s,t) \neq \emptyset\}) \geq d(L'(s,t)) - f(t,s) - p(\{R \in \text{APX}_{\text{mid}} : R \cap L(s,t) \neq \emptyset\})$. Using the equality from Lemma 21, this can be rearranged to

$$p(\{R \in S^+ : R \cap L(s,t) \neq \emptyset\}) - d(L'(s,t)) \geq p(\{R \in S_{\text{right}} : R \cap L(s,t) \neq \emptyset\}) - f(s,t) \qquad (4)$$

Let $\mathcal{H} := \{(c',p') : \exists R \in \bigcup_{W \in \mathcal{W}^{c,p}_{\text{left,top}}} W \text{ with } R \cap L(s,t) \neq \emptyset\}$ denote all (c',p') for which there exists a left-sticking-in row intersecting with the ray. By the second inequality of Lemma 20, we can upper bound

the right hand side of (4) by $\sum_{(c',p')\in\mathcal{H}}(1+\varepsilon)^{p'}\left\lfloor\frac{\varepsilon}{2\cdot K\log T}\bar{n}^{c',p'}\right\rfloor$. Se we have to show that the latter value is also a lower bound for the left hand side.

Intuitively, if we would have S instead of S^+ on the left hand side of (4), we would have a lower bound of 0 for the left hand side. But as S^+ covers substantially more than S on each ray, we will get the desired lower bound. For any group $(c',p')\in\mathcal{H}$, the ray $L(s,t)$ intersects with a rectangle from a row in $\mathcal{W}^{c',p'}_{\text{left,top}}$. So by the last property of Lemma 15 we have

$$\begin{aligned}p(\{R\in S^+\cap\mathcal{R}^{c',p'}_{\text{left}}:R\cap L(s,t)\neq\emptyset\})\geq{}& p(\{R\in S\cap\mathcal{R}^{c',p'}_{\text{left}}:R\cap L(s,t)\neq\emptyset\})\\&+(1+\varepsilon)^{p'}\left\lfloor\frac{\varepsilon}{2\cdot K\log T}\bar{n}^{c',p'}\right\rfloor.\end{aligned}$$

So altogether, we obtain

$$\begin{aligned}p(\{R\in S^+:R\cap L(s,t)\neq\emptyset\})&=p(\{R\in S\setminus\bigcap_{(c',p')\in\mathcal{H}}\mathcal{R}^{c',p'}_{\text{left}}:R\cap L(s,t)\neq\emptyset\})\\&\quad+\sum_{(c',p')\in\mathcal{H}}p(\{R\in S^+\cap\mathcal{R}^{c',p'}_{\text{left}}:R\cap L(s,t)\neq\emptyset\})\\&\geq p(\{R\in S\setminus\bigcap_{(c',p')\in\mathcal{H}}\mathcal{R}^{c',p'}_{\text{left}}:R\cap L(s,t)\neq\emptyset\})\\&\quad+\sum_{(c',p')\in\mathcal{H}}p(\{R\in S\cap\mathcal{R}^{c',p'}_{\text{left}}:R\cap L(s,t)\neq\emptyset\})\\&\quad+\sum_{(c',p')\in\mathcal{H}}(1+\varepsilon)^{p'}\left\lfloor\frac{\varepsilon}{2\cdot K\log T}\bar{n}^{c',p'}\right\rfloor.\\&\geq p(\{R\in S:R\cap L(s,t)\neq\emptyset\})\\&\quad+\sum_{(c',p')\in\mathcal{H}}(1+\varepsilon)^{p'}\left\lfloor\frac{\varepsilon}{2\cdot K\log T}\bar{n}^{c',p'}\right\rfloor.\\&\geq d(L'(s,t))+\sum_{(c',p')\in\mathcal{H}}(1+\varepsilon)^{p'}\left\lfloor\frac{\varepsilon}{2\cdot K\log T}\bar{n}^{c',p'}\right\rfloor.\end{aligned}$$

This yields (4) and thus shows that the ray $L(s,t)$ is covered. So S^+_{left} is a feasible solution for the left subproblem. The proof for S^+_{right} being a feasible solution for the right subproblem is analogous. □

This also completes the proof of Lemma 16.

4.5 Proof of Lemma 18

First, we show that the constructed reference solutions S^+_{left} and S^+_{right} are not too expensive. For that, we use the introduced cost function c_{APX}. Whether a row is centered, left-sticking-in or right-sticking-in depends on the area A of the subproblem, so also c_{APX} depends on A. Therefore, let

$$\begin{aligned}c_{\text{APX}}(A',S'):={}&2\cdot c(S'\cap\mathcal{R}_{\text{center}}(A'))+c(S'\cap\mathcal{R}_{\text{right}}(A'))+c(S'\cap\mathcal{R}_{\text{span}}(A'))\\&+c(\{R\in S'\cap\mathcal{R}_{\text{left}}(A'):\text{left}(R)\leq\text{left}(A')\})+2\cdot c(\{R\in S'\cap\mathcal{R}_{\text{left}}(A'):\text{left}(R)>\text{left}(A')\})\end{aligned}$$

where $\mathcal{R}_{\text{center}}(A')$, $\mathcal{R}_{\text{right}}(A')$, $\mathcal{R}_{\text{span}}(A')$ and $\mathcal{R}_{\text{left}}(A')$ denotes the centered, right-sticking-in, spanning and left-sticking-in rows w.r.t. the area A'. During the algorithm, we redefine the cost of some rectangles in centered rows (i.e., the rectangle $\mathcal{R}_{\text{leftmost}}$) or left-sticking-in rows W (when we treat $W \setminus \text{APX}_{\text{mid}}$ as a centered row), therefore we denote by $c'(R)$ the cost of a rectangle R after redefining the costs.

Lemma 23. *We have* $c_{\text{APX}}(A, S^+) \geq c'_{\text{APX}}(A_{\text{left}}, S^+_{\text{left}}) + c'_{\text{APX}}(A_{\text{right}}, S^+_{\text{right}}) + c(\text{APX}_{\text{mid}})$.

Proof. Let $W \in \mathcal{W}$ be a row and let $R = [\text{left}(R), \text{right}(R)) \times [j, j+1) \in W$ be a rectangle in this row. Furthermore let $W' := \{R' = [\text{left}(R'), \text{right}(R')) \times [j', j'+1) \in \mathcal{R}_{\text{left}} \cup \mathcal{R}_{\text{right}} \cup \text{APX}_{\text{mid}} : j' = j\}$ denote the corresponding rectangles in the left and right subproblem and the selected rectangles. We show that

$$c_{\text{APX}}(A, S^+ \cap W) \geq c'_{\text{APX}}(A_{\text{left}}, S^+_{\text{left}} \cap W') + c'_{\text{APX}}(A_{\text{right}}, S^+_{\text{right}} \cap W') + c(\text{APX}_{\text{mid}} \cap W'). \quad (5)$$

We make a case distinction by the type of row W.

First consider a centered row W. If $W_{\text{left}} = W$ or $W_{\text{right}} = W$ we have $S^+_{\text{left}} \cap W' = W$ or $S^+_{\text{right}} \cap W' = W$, which directly yields (5). So now assume that $W_{\text{left}} \neq W \neq W_{\text{right}}$. Then the row $W_{\text{left}} \cup \{R_{\text{mid,left}}\}$ is right-sticking-in in A_{left} and the row $W_{\text{right}} \cup \{R_{\text{mid,left}}\}$ is left-sticking-in in A_{right}. Suppose that R_{mid} exists and $\text{mid}(W) \in S^+$. Then $S^+_{\text{left}} \cap W' = W_{\text{left}} \cup \{R_{\text{mid,left}}\}$ and $S^+_{\text{right}} \cap W' = S^+ \cap W_{\text{right}} \cup \{R_{\text{mid,right}}\}$. Recall that $c(R_{\text{mid,right}}) = c(W_{\text{left}} \cup \{R_{\text{mid,left}}\})$ and $c(R_{\text{mid,left}}) = c(R_{\text{mid}})$. So

$$\begin{aligned}
&c_{\text{APX}}(A_{\text{left}}, S^+_{\text{left}} \cap W') + c_{\text{APX}}(A_{\text{right}}, S^+_{\text{right}} \cap W') \\
&= c(W_{\text{left}} \cup \{R_{\text{mid,left}}\}) + c(R_{\text{mid,right}}) + 2c(S^+ \cap W_{\text{right}}) \\
&= 2c(W_{\text{left}} \cup \{R_{\text{mid}}\} \cup (S^+ \cap W_{\text{right}})) = 2c(S^+ \cap W) \\
&= c_{\text{APX}}(A, S^+ \cap W)
\end{aligned}$$

And if $\text{mid}(W) \notin S^+$, we have $S_{\text{left}} \cap W' = S^+ \cap W$. This yields $c_{\text{APX}}(S^+ \cap W) = 2c(S^+ \cap W) \geq c(S^+ \cap W) = c_{\text{APX}}(A_{\text{left}}, S_{\text{left}} \cap W_{\text{right}})$ and thus yields (5) as well. So suppose that R_{mid} does not exist and $R_{\text{leftmost}} \in S^+$. Recall that $c'(R_{\text{leftmost}}) = c(R_{\text{leftmost}}) + c(W_{\text{left}})$, which yields

$$\begin{aligned}
&c'_{\text{APX}}(A_{\text{left}}, S^+_{\text{left}} \cap W') + c'_{\text{APX}}(A_{\text{right}}, S^+_{\text{right}} \cap W') \\
&= c(W_{\text{left}}) + c'(R_{\text{leftmost}}) + 2c(S^+ \cap W_{\text{right}} \setminus R_{\text{leftmost}}) \\
&= 2c(W_{\text{left}}) + c(R_{\text{leftmost}}) + 2c(S^+ \cap W_{\text{right}} \setminus R_{\text{leftmost}}) \\
&\leq 2c(W_{\text{left}} \cup (S^+ \cap W_{\text{right}})) = 2c(S^+ \cap W) \\
&= c_{\text{APX}}(A, S^+ \cap W)
\end{aligned}$$

And if $R_{\text{leftmost}} \in S^+$, we again have $S_{\text{left}} \cap W' = S^+ \cap W$ which yields $c_{\text{APX}}(S^+ \cap W) = 2c(S^+ \cap W) \geq c(S^+ \cap W) = c_{\text{APX}}(A_{\text{left}}, S_{\text{left}} \cap W_{\text{right}})$ and thus completes the proof of (5) for centered rows.

Next suppose that W is a right-sticking-in row. Furthermore, suppose that each rectangle in $R \in W \setminus \text{APX}_{\text{mid}}$ satisfies $\text{mid}(A) \leq \text{left}(R)$. Note that $W' \cap R'_{\text{right}}$ is also right-sticking-in. Then $S^+_{\text{left}} \cap W' = (S^+ \cap W) \setminus \text{APX}_{\text{mid}}$ and thus

$$\begin{aligned}
c_{\text{APX}}(A, S^+ \cap W) = c(S^+ \cap W) &= c(\text{APX}_{\text{mid}} \cap W') + c(S^+_{\text{left}} \cap W') \\
&= c'_{\text{APX}}(A_{\text{right}}, S^+_{\text{right}} \cap W') + c(\text{APX}_{\text{mid}} \cap W')
\end{aligned}$$

So suppose that there is a rectangle $R \in W \setminus \text{APX}_{\text{mid}}$ intersecting A_{left}. For such a row $S^+_{\text{left}} \cap W' = S^+ \cap W$ and W_{left} is right-sticking-in in A_{left}. Thus $c_{\text{APX}}(A, S^+ \cap W) = c(S^+ \cap W) = c(S^+_{\text{left}} \cap W') =$

$c_{\mathrm{APX}}(A_{\mathrm{left}}, S^+_{\mathrm{left}} \cap W')$, completing the proof of (5) for right-sticking-in rows. The proof for spanning rows is exactly the same as for right-sticking-in rows.

Finally suppose that W is a left-sticking-in row. Furthermore suppose that each rectangle $R \in W \setminus \mathrm{APX}_{\mathrm{mid}}$ is contained in the interior of A. The same argument as for centered rows shows that $c'_{\mathrm{APX}}(A_{\mathrm{left}}, S^+_{\mathrm{left}} \cap W') + c'_{\mathrm{APX}}(A_{\mathrm{right}}, S^+_{\mathrm{right}} \cap W') \leq 2c(W \setminus \mathrm{APX}_{\mathrm{mid}})$. As W is left-sticking-in there must be a rectangle $R \in W$ with $\mathrm{left}(R) \leq \mathrm{left}(A)$, which is therefore not contained in the interior of A. So $\mathrm{APX}_{\mathrm{mid}} \cap W \neq \emptyset$ and thus $c_{\mathrm{APX}}(A, S^+ \cap W) \geq c(\mathrm{APX}_{\mathrm{mid}} \cap W) + 2c(W \setminus \mathrm{APX}_{\mathrm{mid}})$, which yields (5). So suppose that there is a rectangle $R \in W \setminus \mathrm{APX}_{\mathrm{mid}}$ with $\mathrm{left}(R) \leq \mathrm{left}(A)$. In this case, we either assign all rectangles in $W \setminus \mathrm{APX}_{\mathrm{mid}}$ to the left subproblem or assign them all to the right subproblem. In both cases, the row is left-sticking-in in the respective subproblem. So in both cases

$$\begin{aligned}
&c'_{\mathrm{APX}}(A_{\mathrm{left}}, S^+_{\mathrm{left}} \cap W') + c'_{\mathrm{APX}}(A_{\mathrm{right}}, S^+_{\mathrm{right}} \cap W') + c(\mathrm{APX}_{\mathrm{mid}} \cap W') \\
&\leq c(\{R \in W \setminus \mathrm{APX} : \mathrm{left}(R) \leq \mathrm{left}(A)\}) + 2c(\{R \in W \setminus \mathrm{APX} : \mathrm{left}(R) > \mathrm{left}(A)\}) + c(\mathrm{APX}_{\mathrm{mid}} \cap W') \\
&= c_{\mathrm{APX}}(A, S^+ \cap W)
\end{aligned}$$

This completes the proof of (5) for left-sticking-in rows. And adding up (5) for each row W yields the lemma. □

Now, we show that APX is not more expensive than the selected rectangles $\mathrm{APX}_{\mathrm{mid}}$ and the solutions $\mathrm{APX}_{\mathrm{left}}$ and $\mathrm{APX}_{\mathrm{right}}$ combined.

Lemma 24. *We have* $c(\mathrm{APX}) \leq c'(\mathrm{APX}_{\mathrm{left}}) + c'(\mathrm{APX}_{\mathrm{right}}) + c(\mathrm{APX}_{\mathrm{mid}})$.

Proof. Let $W \in \mathcal{W}$ be a row and let $R = [\mathrm{left}(R), \mathrm{right}(R)) \times [j, j+1) \in W$ be a rectangle in this row. Furthermore let $W' := \{R' = [\mathrm{left}(R'), \mathrm{right}(R')) \times [j', j'+1) \in \mathcal{R}_{\mathrm{left}} \cup \mathcal{R}_{\mathrm{right}} \cup \mathrm{APX}_{\mathrm{mid}} : j' = j\}$ denote the corresponding rectangles in the left and right subproblem and the selected rectangles in $\mathrm{APX}_{\mathrm{mid}}$. We show that $c(\mathrm{APX} \cap W) \leq c(\mathrm{APX}_{\mathrm{left}} \cap W') + c(\mathrm{APX}_{\mathrm{right}} \cap W') + c(\mathrm{APX}_{\mathrm{mid}} \cap W)$. Note that $\mathrm{APX}_{\mathrm{mid}} \cap W$ is always a prefix of row W.

First suppose that the rectangle R_{mid} exists and the algorithm splits this rectangle into $R_{\mathrm{mid,left}}$ and $R_{\mathrm{mid,right}}$, which happens for some centered or left-sticking-in rows. In this case we have $c(R_{\mathrm{mid,left}}) = c(R_{\mathrm{mid}})$ and $c'(R_{\mathrm{mid,right}}) = c(R'_{\mathrm{left}} \cap W')$. So if $\mathrm{APX}_{\mathrm{right}} \cap W' = \emptyset$ and $R_{\mathrm{mid,left}} \notin \mathrm{APX}_{\mathrm{left}}$, we have $\mathrm{APX} \cap W = (\mathrm{APX}_{\mathrm{mid}} \cap W) \cup (\mathrm{APX}_{\mathrm{left}} \cap W')$ and thus $c(\mathrm{APX} \cap W) = c(\mathrm{APX}_{\mathrm{mid}} \cap W) + c(\mathrm{APX}_{\mathrm{left}} \cap W')$. And if $\mathrm{APX}_{\mathrm{right}} \cap W' = \emptyset$ and $R_{\mathrm{mid,left}} \in \mathrm{APX}_{\mathrm{left}}$, we have $\mathrm{APX} \cap W = (\mathrm{APX}_{\mathrm{mid}} \cap W) \cup (\mathrm{APX}_{\mathrm{left}} \cap W') \setminus \{R_{\mathrm{mid,left}}\} \cup \{R_{\mathrm{mid}}\}$ and thus $c(\mathrm{APX} \cap W) = c(\mathrm{APX}_{\mathrm{mid}} \cap W) + c(\mathrm{APX}_{\mathrm{left}} \cap W') - c(R_{\mathrm{mid,left}}) + c(R_{\mathrm{mid}}) = c(\mathrm{APX}_{\mathrm{mid}} \cap W) + c(\mathrm{APX}_{\mathrm{left}} \cap W')$. And if $\mathrm{APX}_{\mathrm{right}} \cap W' \neq \emptyset$ we have $\mathrm{APX} \cap W = (\mathrm{APX}_{\mathrm{mid}} \cap W) \cup (\mathcal{R}'_{\mathrm{left}} \cap W') \cup (\mathrm{APX}_{\mathrm{right}} \cap W') \setminus \{R_{\mathrm{mid,left}}, R_{\mathrm{mid,right}}\} \cup \{R_{\mathrm{mid}}\}$ and thus

$$\begin{aligned}
c(\mathrm{APX} \cap W) &= c(\mathrm{APX}_{\mathrm{mid}} \cap W) + c(\mathcal{R}'_{\mathrm{right}} \cap W' \setminus \{R_{\mathrm{mid,left}}\}) + c(\mathrm{APX}_{\mathrm{right}} \cap W' \setminus \{R_{\mathrm{mid,right}}\}) + c(R_{\mathrm{mid}}) \\
&= c(\mathrm{APX}_{\mathrm{mid}} \cap W) + c'(R_{\mathrm{mid,right}}) + c(\mathrm{APX}_{\mathrm{right}} \cap W' \setminus \{R_{\mathrm{mid,right}}\}) \\
&= c(\mathrm{APX}_{\mathrm{mid}} \cap W) + c'(\mathrm{APX}_{\mathrm{right}} \cap W')
\end{aligned}$$

This yields the lemma in the case where R_{mid} exists and the algorithm splits this rectangle into $R_{\mathrm{mid,left}}$ and $R_{\mathrm{mid,right}}$.

So suppose that this is not the case. Note that by construction we have $W = (\mathrm{APX}_{\mathrm{mid}} \cap W') \cup (\mathcal{R}'_{\mathrm{left}} \cap W') \cup (\mathcal{R}'_{\mathrm{right}} \cap W')$. If $\mathrm{APX}_{\mathrm{right}} \cap W' = \emptyset$ then $\mathrm{APX} \cap W = (\mathrm{APX}_{\mathrm{mid}} \cap W') \cup (\mathrm{APX}_{\mathrm{left}} \cap W')$, which yields the lemma. So suppose that $\mathrm{APX}_{\mathrm{right}} \cap W' \neq \emptyset$ and let R be the leftmost rectangle of $\mathrm{APX}_{\mathrm{right}}$. Then

R is also the leftmost rectangle of $\mathcal{R}'_{\text{right}} \cap W'$ and by construction we have $c'(R) = c(R) + c(\mathcal{R}'_{\text{left}} \cap W')$. So

$$\begin{aligned} c(\text{APX} \cap W) &= c(\text{APX}_{\text{mid}} \cap W') + c(\mathcal{R}'_{\text{left}} \cap W')c(\text{APX}_{\text{right}} \cap W') \\ &= c(\text{APX}_{\text{mid}} \cap W') + c'(\text{APX}_{\text{right}} \cap W') \end{aligned}$$

which completes the proof. □

Now we can combine Lemmas 15,23 and 24 to obtain the following approximation ratio.

Lemma 25. *Let S be any feasible solution for a subproblem with region A. Then*

$$c(\text{APX}) \leq \left(1 + \frac{\varepsilon \log(\text{right}(A) - \text{left}(A))}{\log T}\right) c_{\text{APX}}(A, S)$$

Proof. We prove the lemma by induction on $b - a$. If $b - a = 1$, the problem can be solved exactly according to Lemma 13. So suppose $b - a > 1$. Then S^+_{left} and S^+_{right} are feasible solutions for the left and right subproblem. By the induction hypothesis, we have

$$c(\text{APX}_{\text{left}}) \leq \left(1 + \frac{\varepsilon \log((b-a)/2)}{\log T}\right) c_{\text{APX}}(A_{\text{left}}, S^+_{\text{left}})$$

$$c(\text{APX}_{\text{right}}) \leq \left(1 + \frac{\varepsilon \log((b-a)/2)}{\log T}\right) c_{\text{APX}}(A_{\text{right}}, S^+_{\text{right}})$$

Together with Lemmas 24 and 23 we obtain

$$\begin{aligned} c(\text{APX}) &\leq c(\text{APX}_{\text{left}}) + c(\text{APX}_{\text{right}}) + c(\text{APX}_{\text{mid}}) \\ &\leq \left(1 + \frac{\varepsilon \log((b-a)/2)}{\log T}\right) \cdot \left(c_{\text{APX}}(A_{\text{left}}, S^+_{\text{left}}) + c_{\text{APX}}(A_{\text{right}}, S^+_{\text{right}}) + c(\text{APX}_{\text{mid}})\right) \\ &\leq \left(1 + \frac{\varepsilon \log((b-a)/2)}{\log T}\right) \cdot c_{\text{APX}}(A, S^+) \\ &\leq \left(1 + \frac{\varepsilon(\log(b-a) - 1)}{\log T}\right) \cdot \left(1 + \frac{\varepsilon}{2 \log T}\right) c_{\text{APX}}(A, S) \\ &\leq \left(1 + \frac{\varepsilon \log(b-a)}{\log T}\right) \cdot c_{\text{APX}}(A, S) \end{aligned}$$

In the second to last step, we used Lemma 15. This completes the proof. □

This directly the approximation ratio stated in Lemma 18.

Proof of Lemma 18. Recall that for the root subproblem $A = [0, T) \times [0, \infty)$. By Lemma 25, we know that $c(\text{APX}^{\text{root}}) \leq \left(1 + \frac{\varepsilon \log(\text{right}(A) - \text{left}(A))}{\log T}\right) c_{\text{APX}}(A, S) \leq (1 + \varepsilon) c_{\text{APX}}(A, S) \leq (2 + 2\varepsilon) c(S)$. □

5 Weighted Tardiness

In this section prove Theorem 2, i.e., we present a $(1+\varepsilon)$-approximation algorithm for the weighted tardiness objective for any constant $\varepsilon > 0$ with a running time of $2^{\mathrm{poly}(\log(n+p_{\max}))}$. Recall that in this setting each job has a due date $d_j \in \mathbb{N}$ and a weight $w_j \in \mathbb{N}$ and its cost function is $\mathrm{cost}_j(t) = w_j(t - d_j)$ for each $t \in \mathbb{R}$. First, we reduce the problem to a special case of RCP in which each instance has certain structural properties. Then we give a recursive algorithm which is a $(1 + \varepsilon)$-approximation algorithm for this setting of RCP. In particular, due to the additional structural properties this algorithm is much simpler than the algorithm for the general case of RCP (see Section 4.2) and at the same time achieves an approximation ratio of only $1 + \varepsilon$, rather than $2 + \varepsilon$.

Formally, in our special case of RCP, each instance, given by a set of rectangles $\mathcal{R}$ and a set of rays $\mathcal{L}$, is *δ-well-structured* (for some $\delta > 0$ that our running time will depend on) which means that for each rectangle $R = [\mathrm{left}(R), \mathrm{right}(R)) \times [j, j+1) \in \mathcal{R}$ and the (unique) value $h \in \mathbb{N}$ with $2^h \leq \mathrm{right}(R) - \mathrm{left}(R) < 2^{h+1}$ we have that $\mathrm{left}(R)$ and $\mathrm{right}(R)$ are integral multiples of $\delta \cdot 2^h$. We reduce the weighted tardiness problem to this special case of RCP in the following lemma, which we will prove in Section 5.1.

In the following, assume without loss of generality that $\varepsilon = 2^{-k}$ for some $k \in \mathbb{N}$.

Lemma 26. *Given an α-approximation algorithm for δ-well-structured instances of RCP with a running time of $f(n, K, \max_R \mathrm{right}(R), p_{\max}, \delta)$, there is a $\alpha(1 + \varepsilon)$-approximation algorithm for minimizing weighted tardiness in time $f((np_{\max})^{O(1)}, (1/\varepsilon)^{O(1/\varepsilon^3)}, (np_{\max})^{O(1)}, \varepsilon/32)$.*

Using this reduction, it suffices to solve δ-well-structured instances of RCP. We will present an algorithm for this task, corresponding to the following lemma, in Section 5.2.

Lemma 27. *There is a $(1 + \varepsilon)$-approximation algorithm for δ-well-structured instances of RCP with a running time of $2^{(1/\delta \cdot K \cdot \log(n \cdot p_{\max} \cdot \max_R \mathrm{right}(R))/\varepsilon)^{O(K)}}$.*

Together, Lemmas 26 and 27 yield Theorem 2.

5.1 Reduction

Recall that the x-coordinates for the rectangles in RCP are derived from the milestones from Theorem 7. We modify the construction here, achieving the same properties, but additionally an equivalent of δ-well-structuredness. Recall also that $T = O(\max_{j \in J} r_j + \sum_{j \in J} p_j)$ is an upper bound on the time when the last job is finished.

Lemma 28. *Consider an instance of weighted tardiness minimization. For each job j we can in polynomial time construct a sequence $m_0(j), m_1(j), m_2(j), \ldots, m_{f_j}(j) \in \mathbb{N}$ where*

1. $\mathrm{cost}_j(m_{i+1}(j)) \leq (1 + \varepsilon) \cdot \mathrm{cost}_j(m_i(j) + 1)$ *for all* $0 \leq i < f_j$*;*
2. $\mathrm{cost}_j(m_{i+1}(j) + 1) > (1 + \varepsilon/4) \cdot \mathrm{cost}_j(m_i(j) + 1)$ *for all* $0 \leq i < f_j - 1$*;*
3. $m_0(j) = r_j$*;*
4. $m_1(j) = d_j$*;*
5. $m_{f_j}(j) = T$*;*
6. *let $h \in \mathbb{N}$ with $2^h \leq m_{i+1}(j) - m_i(j) < 2^{h+1}$. Then $m_i(j), m_{i+1}(j) \in \delta 2^h \cdot \mathbb{Z}$, where $\delta = \varepsilon/32$.*

Proof. We set $m_0(j) = r_j$ and $m_1(j) = d_j$. Then for $i \geq 1$ suppose we have already defined $m_i(j)$. Let $h' \in \mathbb{N}$ with $2^{h'} \leq m_i(j) - d_j < 2^{h'+1}$. We define $m_{i+1}(j) \in \{m_i(j)+1, m_i(j)+2, \ldots, T\}$ to be the minimal number that is at least as large as the *second* smallest $t \in \frac{\varepsilon}{2}2^{h'} \cdot \mathbb{Z}$ with $t > m_i(j)$; or T if this does not exists (because it would exceed T). Let $f_j \in \mathbb{N}$ be the smallest index i where $m_i(j)$ equals T.

It is easy to check that Properties 1 and 2 hold for $i = 0$. Let $1 \leq i < f_j$. Then

$$\begin{aligned}\mathrm{cost}_j(m_{i+1}(j)) = w_j(m_{i+1}(j) - d_j) &\leq w_j(m_i(j) + 1 + \varepsilon 2^{h'} - d_j) \\ &\leq (1+\varepsilon)(m_i(j) + 1 - d_j) = (1+\varepsilon)\mathrm{cost}_j(m_i(j)+1).\end{aligned}$$

Furthermore, for $1 \leq i < f_j - 1$,

$$\begin{aligned}\mathrm{cost}_j(m_{i+1}(j)+1) = w_j(m_{i+1}(j) + 1 - d_j) &> w_j(m_i(j) + 1 + \varepsilon 2^{h'-1} - d_j) \\ &\geq (1+\varepsilon/4)(m_i(j) + 1 - d_j) = (1+\varepsilon/4)\mathrm{cost}_j(m_i(j)+1).\end{aligned}$$

Consider now Property 6. If $m_{i+1}(j) - m_i(j) \leq 32/\varepsilon$ then the claim is trivial, since $\varepsilon/32 \cdot 2^h \leq 1$, and $m_i(j), m_{i+1}(j) \in \mathbb{Z}$. Hence, assume otherwise. Let $h \in \mathbb{N}$ with $2^h \leq m_{i+1}(j) - m_i(j) < 2^{h+1}$. It holds that that $m_i(j) - d_j < 2^h$, since otherwise $m_{i+1}(j) < m_i(j) + \varepsilon 2^h$, a contradiction to the definition of h. It follows that $m_{i+1}(j) \in \frac{\varepsilon}{2}2^h \cdot \mathbb{Z} \subseteq \frac{\varepsilon}{32}2^h \cdot \mathbb{Z}$.

For the discreteness of $m_i(j)$, we will argue that the differences $m_{i+1}(j) - m_i(j)$ for $i > 1/\varepsilon$ do not grow too quickly. Note that we have

$$m_{i+1}(j) - m_i(j) > \varepsilon(m_i(j) - d_j)/4.$$

Furthermore, for all $i > 2/\varepsilon$ we have

$$m_{i+1}(j) - m_i(j) \leq 2\varepsilon(m_i(j) - d_j).$$

Assuming that ε is sufficiently small it follows that

$$m_{i+1}(j) - d_j \leq 2(m_i(j) - d_j).$$

Thus,

$$m_{i+1}(j) - m_i(j) \leq 2\varepsilon(m_i(j) - d_j) \leq 4\varepsilon(m_{i-1} - d_j) < 16(m_i(j) - m_{i-1}(j)).$$

It follows that $2^{h''} \leq m_i(j) - m_{i-1}(j)$, where $2^{h''} \geq 2^h/16$. By the argument used to show $m_{i+1}(j) \in \frac{\varepsilon}{2}2^h \cdot \mathbb{Z}$ we obtain $m_i(j) \in \frac{\varepsilon}{2}2^{h''} \cdot \mathbb{Z} \subseteq \frac{\varepsilon}{32}2^h \cdot \mathbb{Z}$. □

Since this construction satisfies all properties of Theorem 7, we can use it with the reduction in Section 3. Recall that the rectangles $[a, b) \times [j', j'+1) \in \mathcal{R}$ all had the form that there is some $j \in J$ and i with $a = m_i(j)$ and $b = m_{i+1}(j)$. Assuming that $i \geq 1$, the previous lemma ensures that for $h \in \mathbb{N}$ with $2^h \leq b - a < 2^{h+1}$ we have $a, b \in \frac{\varepsilon}{32}2^h \cdot \mathbb{Z}$. The first rectangle of each job with x-coordinates $[m_0(j), m_1(j)) = [r_j, d_j)$ may be very wide and r_j, d_j cannot be rounded. This rectangle, however, has cost zero, so the reduction in Section 3 anyway removes it in the preprocessing. The instance size and K are bounded as in Theorem 3. The maximum right hand side of a rectangle, i.e., $\max_R \mathrm{right}(R)$, is bounded by T and T is bounded by $(n + p_{\max})^{O(1)}$ via the preprocessing in Section 3. Hence, this implies Theorem 26.

5.2 Algorithm

In this section we prove Lemma 27. Suppose that we are given a δ-well-structured instance of RCP. Our approach to solve it is similar to our algorithm for general case of RCP. As before, let $p_{\max} = \max_R p(R)$. We start with a preprocessing step. This time, we argue that it suffices to construct an algorithm for which $C := \frac{\max_{R\in\mathcal{R}} c(R)}{\min_{R\in\mathcal{R}} c(R)} = O(K \cdot n/\varepsilon)$.

Lemma 29. *Assume that there is an α-approximation algorithm for δ-well-structured instances of RCP with a running time of $2^{(1/\delta\cdot\log C\cdot K\cdot\log(n\cdot p_{\max}\cdot\max_R \operatorname{right}(R)))^{O(K)}}$. Then there is an $(1+\varepsilon)\alpha$-approximation algorithm for δ-well-structured instances of RCP with a running time of $2^{(1/\delta\cdot K\cdot\log(n\cdot p_{\max}\cdot\max_R \operatorname{right}(R))/\varepsilon)^{O(K)}}$.*

Proof. Let $\mathcal{R}$, $\mathcal{L}$ denote the rectangles and rays of a δ-well-structured instance I and let S be an optimal solution to this instance. Then we guess the most expensive rectangle $R_{\max} \in S$ in the optimal solution. Let $\mathcal{W}^{\text{cheap}}$ denote all rows in which there exists a rectangle of cost at most $\varepsilon \cdot c(R_{\max})/(n \cdot K)$. Let $\mathcal{R}^{\text{cheap}}$ denote all rectangles in the rows $\mathcal{W}^{\text{cheap}}$. We select $\mathcal{R}^{\text{cheap}}$. As for each row $W \in \mathcal{W}^{\text{cheap}}$ we have $\sum_{R\in W} c(R) \leq K \cdot \varepsilon \cdot c(R_{\max})/(n \cdot K) \leq \varepsilon c(R_{\max})/n$, this implies $c(\mathcal{R}^{\text{cheap}}) \leq \varepsilon c(R_{\max})$.

Let $\mathcal{R}^{\text{discard}}$ denote all rectangles R with $c(R) > c(R_{\max})$ and all rectangles in the same row on the right of such a rectangle. Note that the optimal solution cannot select any rectangle in $\mathcal{R}^{\text{discard}}$ as it contains a prefix from each row. Next, we define a set of rays $\mathcal{L}'$. For each ray $L(s,t) \in \mathcal{L}$ we add a ray $L'(s,t) \in \mathcal{L}'$ with a demand of $d(L'(s,t)) = \max\{0, d(L(s,t)) - p(\{R \in \mathcal{R}^{\text{cheap}} : R \cap L(s,t) \neq \emptyset\})\}$. Then we use the given algorithm for the instance I' with rectangles $\mathcal{R} \setminus (\mathcal{R}^{\text{cheap}} \cup \mathcal{R}^{\text{discard}})$ and the rays $\mathcal{L}'$. Given a solution APX to this instance, we output $\text{APX} \cup \mathcal{R}^{\text{cheap}}$. The most expensive rectangle in I' has a cost of $c(R_{\max})$ and the cheapest rectangle has a cost of at least $\varepsilon \cdot c(R_{\max})/(n \cdot K)$, so we have $C \leq K \cdot n/\varepsilon$ for the instance I'.

Clearly $\text{APX} \cup \mathcal{R}^{\text{cheap}}$ is a feasible solution for the instance I. Also $S \setminus \mathcal{R}^{\text{cheap}}$ is a feasible solution for I'. So $c(\text{APX}) \leq \alpha \cdot c(S \setminus \mathcal{R}^{\text{cheap}})$, which implies $c(\text{APX} \cup \mathcal{R}^{\text{cheap}}) \leq \alpha \cdot c(S \setminus \mathcal{R}^{\text{cheap}}) + \varepsilon c(R_{\max}) \leq c(S \setminus \mathcal{R}^{\text{cheap}}) + \varepsilon c(S) \leq (1+\varepsilon)\alpha c(S)$. This shows that we obtain a $(1+\varepsilon)\alpha$-approximation. As we need to call the given algorithm at most n times (once for each choice of $R_{\max}$) and $C \leq K \cdot n/\varepsilon$ for the instance I', we obtain the claimed running time. □

Our algorithm is based on a recursion in which the input of each recursive call consists of

- an instance of RCP defined by a set of rays $\mathcal{L}'$ and a set of rectangles $\mathcal{R}'$; we denote by $\mathcal{W}'$ be the partition of $\mathcal{R}'$ into rows,
- an area A of the form $A = [\operatorname{left}(A), \operatorname{right}(A)) \times [0, \infty)$ for two values $\operatorname{left}(A), \operatorname{right}(A) \in \mathbb{N}_0$ such that $\operatorname{right}(A) - \operatorname{left}(A) = 2^k$ for some $k \in \mathbb{N}_0 \cup \{-1\}$ and $\operatorname{left}(A)$ and $\operatorname{right}(A)$ are integral multiples of 2^k,
- each rectangle $R \in \mathcal{R}'$ is contained in A, i.e., $R \subseteq A$,
- each ray $L \in \mathcal{L}'$ is contained in $[\operatorname{left}(A), \operatorname{right}(A)) \times \mathcal{R}$.

The recursive call returns a solution to the RCP instance defined by $\mathcal{R}'$ and $\mathcal{L}'$. At the end of our algorithm, we output the solution returned by the (main) recursive call in which $\mathcal{R}' := \mathcal{R}$, $\mathcal{L}' := \mathcal{L}$, $\operatorname{left}(A) = 0$, and $\operatorname{right}(A) := T$ where T is the smallest power of 2 that is at least $\max_{R\in\mathcal{R}} \operatorname{right}(R)$. Any solution to this subproblem is a solution to our given instance.

Assume we are given a recursive call as defined above. The base case of our recursion arises when $\operatorname{right}(A) - \operatorname{left}(A) = 1/2$. Then $\mathcal{R}' = \emptyset$ since by assumption each rectangle in $\mathcal{R}'$ has integer coordinates

and is contained in A. Therefore, the subproblem is feasible if an only if for each ray $L \in \mathcal{L}'$ we have that $d(L) = 0$. Assume now that $\mathrm{right}(A) - \mathrm{left}(A) \geq 1$. As for the general case of RCP, we split the problem into a left and a right subproblem at the vertical line $\mathrm{mid}(A) \times \mathbb{R}$ where $\mathrm{mid}(A) := (\mathrm{right}(A) - \mathrm{left}(A))/2$. Formally, the areas for the left and right subproblem are defined by $A_{\mathrm{left}} := [\mathrm{left}(A), \mathrm{mid}(A)) \times [0, \infty)$ and $A_{\mathrm{right}} := [\mathrm{mid}(A), \mathrm{right}(A)) \times [0, \infty)$, respectively. In the following, we will define the rectangles $\mathcal{R}'_{\mathrm{left}}$ and $\mathcal{R}'_{\mathrm{right}}$ for the left and right subproblem, respectively, and select certain rectangles from $\mathcal{R}' \setminus (\mathcal{R}'_{\mathrm{left}} \cup \mathcal{R}'_{\mathrm{right}})$. Finally define the rays $\mathcal{L}'_{\mathrm{left}}$ and $\mathcal{L}'_{\mathrm{right}}$ for the left and right subproblem, respectively,

First, we define that $\mathcal{R}'_{\mathrm{left}}$ contains all rectangles from each row $W \in \mathcal{W}'$ in which *each* rectangle $R \in W$ is contained in A_{left}, i.e., $R \subseteq A_{\mathrm{left}}$. Symmetrically, we define that $\mathcal{R}'_{\mathrm{right}}$ contains all rectangles from each row $W \in \mathcal{W}'$ in which each rectangle $R \in W$ is contained in A_{right}, i.e., $R \subseteq A_{\mathrm{right}}$. Consider now all remaining rows in $\mathcal{W}'$, i.e., let $\mathcal{W}^{\mathrm{cross}} \subseteq \mathcal{W}'$ be the set of all rows $W \in \mathcal{W}'$ which contain a rectangle $R \in W$ with $R \cap A_{\mathrm{left}} \neq \emptyset$ and a rectangle $R' \in W$ with $R \cap A_{\mathrm{right}} \neq \emptyset$.

We partition the rows in $\mathcal{W}^{\mathrm{cross}}$ into groups. Intuitively, the rectangles of rows in the same group are vertical translates of each other, have approximately the same cost, and the same value. Formally, a group $\mathcal{W}^{\mathrm{cross}}_g \subseteq \mathcal{W}^{\mathrm{cross}}$ is specified by a tuple $g = (c_1, \dots c_k, p, t_0, \dots, t_k)$ of integers where $k \leq K$ (recall that K is an upper bound on the number of rectangles in a row). Consider a row $W \in \mathcal{W}^{\mathrm{cross}}$ and assume that $R_1, \dots R_{k'}$ are its rectangles such that $\mathrm{left}(R_i) < \mathrm{left}(R_{i+1})$ for each $i \in [k' - 1]$. We define that the row W is in group $\mathcal{W}^{\mathrm{cross}}_g$ if and only if

- $k = k'$,
- $(1+\varepsilon)^p \leq p(R_1) < (1+\varepsilon)^{p+1}$, and
- for $i \leq k$ we have that $(1+\varepsilon)^{c_i} \leq c(R_i) \leq (1+\varepsilon)^{c_i+1}$, $\mathrm{left}(R_i) = t_{i-1}$ and $\mathrm{right}(R_i) = t_i$.

Let $\mathcal{G} := \{g \in \mathbb{Z}^4 \cup \mathbb{Z}^6 \cup \dots \cup \mathbb{Z}^{2K+2} : \mathcal{W}^{\mathrm{cross}}_g \neq \emptyset\}$ denote all vectors g specifying a non-empty group $\mathcal{W}^{\mathrm{cross}}_g$. Next, we show that there is only a quasi-polynomial number of groups in $\mathcal{G}$.

Lemma 30. *We have that* $|\mathcal{G}| \leq (1/\delta \cdot \log C \cdot \log n \cdot \log p_{\max} \cdot \log T/\varepsilon)^{O(K)}$.

Proof. Let $g \in \mathcal{G}$ and consider a row $W \in \mathcal{W}^{\mathrm{cross}}_g$. Let $R_{\mathrm{mid}} \in W$ be the rectangle intersecting $\mathrm{mid}(A) \times \mathbb{R}$, i.e. with $\mathrm{left}(R) \leq \mathrm{mid}(A)$ and $\mathrm{right}(R) > \mathrm{mid}(A)$. We can limit the number of options for g as follows:

- There are K options for the number k of rectangles in row W.
- Recall that $C = \frac{\max_{R \in \mathcal{R}} c(R)}{\min_{R \in \mathcal{R}} c(R)}$. So there are only only $O((\log C)/\varepsilon)$ different integers c_i for which there exists a rectangle $R \in \mathcal{R}$ with $(1+\varepsilon)^{c_i} \leq c(R) \leq (1+\varepsilon)^{c_i+1}$. Thus there are $O(((\log C)/\varepsilon)^K)$ options for $c_1, \dots, c_k$.
- There are $O((\log p_{\max})/\varepsilon)$ options for p as $p(R) \leq p_{\max}$ for all $R \in \mathcal{R}$.
- The length of any rectangle $R \in W$ is at most T. So there are $\log T$ possible values of h such that $2^h \leq \mathrm{right}(R) - \mathrm{left}(R) \leq 2^{h+1}$. As the instance is δ-well-structured, there are $O((\log T)/\delta)$ possible lengths for any rectangle. So there are $O((\log T)/\delta)$ options for the length of R_{mid}.
- The rectangle R_{mid} fulfills $\mathrm{left}(R) \leq \mathrm{mid}(A)$ and $\mathrm{right}(R) > \mathrm{mid}(A)$ and as the instance is δ-well-structured, there are only $1/\delta$ options for $\mathrm{left}(R_{\mathrm{mid}})$ when the length of R_{mid} is fixed.
- There are k options which of the rectangles $R_1, \dots, R_k$ is R_{mid}.

- The rectangles in a row are adjacent. So given the exact position for one rectangle (in this case R_{mid}), it suffices to enumerate all possible lengths of the other rectangles, as this determines their position. As there are $O(\log T/\delta)$ options for the length of a single rectangle, there are $O((\log T/\delta)^{K-1})$ options for the rectangles $W \setminus \{R_{\text{mid}}\}$.

So it suffices to know $k, c_1, \dots, c_k, p$, the lengths of the rectangles $R_1, \dots, R_k$, the position $\text{left}(R_{\text{mid}})$ and the information which of the rectangles $R_1, \dots, R_k$ is R_{mid} to determine the group. So $|\mathcal{G}| = O(K \cdot ((\log C)/\varepsilon)^K \cdot (\log p_{\max})/\varepsilon \cdot ((\log T)/\delta)^K \cdot 1/\delta \cdot K) = (1/\delta \cdot \log C \cdot \log p_{\max} \cdot (\log T)/\varepsilon)^{O(K)}$. □

Let S be an optimal solution to the given subproblem. Intuitively, we want to guess the rectangles in S in each row $\mathcal{W}_g^{\text{cross}}$ for some $g \in \mathcal{G}$. Consider a group $\mathcal{W}_g^{\text{cross}}$ for some $g \in \mathcal{G}$. Essentially, the main difference of the rows within a group is their y-coordinate, since for the rectangles in such a row the costs and the values are almost the same by the definition of the groups. When two rectangles differ only by their y-coordinate, every ray $L \in \mathcal{L}'$ intersecting with the top rectangle also intersects with the bottom one. Therefore, intuitively it is sufficient to select the same number of rectangles from the group g as S and select them greedily from bottom to top, if we select by a factor $1 + \varepsilon$ more rectangles to compensate for the fact that their values may differ by a factor $1 + \varepsilon$.

Formally for the group g, we do the following: For each $0 \leq i \leq k$ let $\mathcal{W}_{g,i}^{\text{cross}}$ denote the rows in $\mathcal{W}_g^{\text{cross}}$ from which S contains exactly i rectangles. Let $n_{g,i} := |\mathcal{W}_{g,i}^{\text{cross}}|$ denote the number of such rows. First, we guess the number $n_{g,i}$ for each $i \in [K]$. For each $i \in [K]$ with $n_{g,i} \leq 1/\varepsilon$ we simply guess the corresponding rows $\mathcal{W}_{g,i}^{\text{cross}}$ and for each row $W \in \mathcal{W}_{g,i}^{\text{cross}}$ we select its leftmost i rectangles. Let $\text{APX}_{g,i}$ denote the resulting rectangles. Let $\mathcal{W}_g^{\text{done}} = \bigcup_{i \in [k]: n_{g,i} \leq 1/\varepsilon} \mathcal{W}_{g,i}^{\text{cross}}$ denote all rows, from which we already guessed the rectangles in S.

We go through the indices i with $n_{g,i} \geq 1/\varepsilon$ in decreasing order, i.e., from K to 1. Consider an index $i \in [K]$. Intuitively, in the $\lfloor (1 + 2\varepsilon) n_{g,i} \rfloor$ bottom rows where no rectangle is selected yet, we add the first i rectangles to our solution. Formally, let $\mathcal{W}_{g,i}^*$ denote the bottom-most $\lfloor (1 + 2\varepsilon) n_{g,i} \rfloor$ rows in $\mathcal{W}_g \setminus (\mathcal{W}_g^{\text{done}} \cup \bigcup_{i' : i < i' \leq K, n_{g,i'} > 1/\varepsilon} \mathcal{W}_{g,i'}^*)$. Now let $\text{APX}_{g,i}$ denote all rectangles that are within the first i rectangles in some row $W \in \mathcal{W}_{g,i}^*$. We also select $\text{APX}_{g,i}$. We define $\text{APX}_g := \bigcup_{1 \leq i \leq k} \text{APX}_{g,i}$ and $S_g := \bigcup_{W \in \mathcal{W}_g} S \cap W$. In the following lemma we prove that APX_g is not much more expensive than S_g.

Lemma 31. *We have that* $c(\text{APX}_g) \leq (1 + 5\varepsilon) c(S_g)$.

Proof. Let $g = (c_1, \dots c_k, p, t_0, \dots, t_k) \in \mathcal{G}$. By construction we have $\text{APX}_g \cap \bigcup_{W \in \mathcal{W}_g^{\text{done}}} = S_g \cap \bigcup_{W \in \mathcal{W}_g^{\text{done}}} W$. So we can intuitively ignore the rows $\mathcal{W}_g^{\text{done}}$. Furthermore for each $i \leq k$ with $n_{g,i} > 1/\varepsilon$ we have that $\text{APX}_{g,i}$ contains the first i rectangles from $\lfloor (1 + 2\varepsilon) n_{g,i} \rfloor$ rows. Due to the definition of g, we have that the cost of the first i rectangles in g is at least $\sum_{i' \leq i} (1 + \varepsilon)^{c_{i'}}$ and at most $\sum_{i' \leq i} (1 + \varepsilon)^{c_{i'}+1}$. So $c(\text{APX}_{g,i}) \leq (1 + 2\varepsilon) n_{g,i} \sum_{i' \leq i} (1 + \varepsilon)^{c_{i'}+1}$ and $c(S \cap \bigcup_{W \in \mathcal{W}_{g,i}^{\text{cross}}} W) \geq n_{g,i} \sum_{i' \leq i} (1 + \varepsilon)^{c_{i'}}$. This implies

$c(\mathrm{APX}_{g,i}) \leq (1+\varepsilon)(1+2\varepsilon)c(S \cap \bigcup_{W \in \mathcal{W}^{\mathrm{cross}}_{g,i}} W) \leq (1+5\varepsilon)c(S \cap \bigcup_{W \in \mathcal{W}^{\mathrm{cross}}_{g,i}} W)$. So we obtain

$$\begin{aligned}
c(\mathrm{APX}_g) &\leq c(\mathrm{APX}_g \cap \bigcup_{W \in \mathcal{W}^{\mathrm{done}}_g}) + \sum_{i \in [k] : n_{g,i} > 1/\varepsilon} c(\mathrm{APX}_{g,i}) \\
&\leq c(S_g \cap \bigcup_{W \in \mathcal{W}^{\mathrm{done}}_g}) + \sum_{i \in [k] : n_{g,i} > 1/\varepsilon} (1+5\varepsilon) c(S \cap \bigcup_{W \in \mathcal{W}^{\mathrm{cross}}_{g,i}} W) \\
&\leq (1+5\varepsilon) c(\sum_{i \leq k} \mathcal{W}^{\mathrm{cross}}_{g,i}) \\
&= (1+5\varepsilon) c(S_g)
\end{aligned}$$

This completes the proof. □

Next, we also show that it covers the same amount on any ray as S_g, so it is a good substitute for S_g.

Lemma 32. *For any ray $L(s,t) \in \mathcal{L}'$, we have that $\sum_{R \in \mathrm{APX}_g : R \cap L(s,t) \neq \emptyset} p(R) \geq \sum_{R \in S_g : R \cap L(s,t) \neq \emptyset} p(R)$.*

Proof. Let $g = (c_1, \dots c_k, p, t_0, \dots, t_k) \in \mathcal{G}$. By construction we have $\mathrm{APX}_g \cap \bigcup_{W \in \mathcal{W}^{\mathrm{done}}_g} = S_g \cap \bigcup_{W \in \mathcal{W}^{\mathrm{done}}_g} W$. So we can intuitively ignore the rows $\mathcal{W}^{\mathrm{done}}_g$. Let $L(s,t) \in \mathcal{L}'$ be a ray. In each row $W \in \mathcal{W}^{\mathrm{done}}_g$ the sets APX_g and S_g contain the same rectangles, so they also cover the same amount on $L(s,t)$. Suppose that there is no row $W \in \mathcal{W}_g \setminus \mathcal{W}^{\mathrm{done}}_g$ with a rectangle $R \in W \setminus \mathrm{APX}_g$ and $R \cap L(s,t) \neq \emptyset$. Then every rectangle $\{R \in S_g : R \cap L(s,t) \neq \emptyset\}$ is also in APX_g, which yields the lemma. So suppose now that there is a row $W' \in \mathcal{W}_g \setminus \mathcal{W}^{\mathrm{done}}_g$ such that there exists a rectangle $R' \in W' \setminus \mathrm{APX}_g$ with $R \cap L(s,t) \neq \emptyset$. Let $i' \leq k$ such that $t_{i'-1} \leq t < t_{i'}$. Then $\mathrm{left}(R') = t_{i'-1}$ and $\mathrm{right}(R') = t_{i'}$. For each $i \geq i'$, we have $W' \notin \mathcal{W}^*_{g,i}$ as APX_g contains the first i rectangles from each row in $\mathcal{W}^*_{g,i}$, but not the i-th rectangle R' in row W'. Recall that $\mathcal{W}^*_{g,i}$ denotes the bottom-most $\lfloor (1+2\varepsilon) n_{g,i} \rfloor$ rows in $\mathcal{W}_g \setminus (\mathcal{W}^{\mathrm{done}}_g \cup \bigcup_{i' : i < i' \leq K, n_{g,i'} > 1/\varepsilon} \mathcal{W}^*_{g,i'})$. This implies that all rows in $\mathcal{W}^*_{g,i}$ are below W'. As the i'-th rectangle in each such row is in APX_g and intersects $L(s,t)$, we have $|\{R \in \mathrm{APX}_g \setminus \bigcup_{W \in \mathcal{W}^{\mathrm{done}}_g} W : R \cap L(s,t) \neq \emptyset\}| \geq \sum_{i \in \{i', \dots, k\} : n_{g,i} > 1/\varepsilon} \lfloor (1+2\varepsilon) n_{g,i} \rfloor$. By definition of $n_{g,i}$ we have $|\{R \in S_g \setminus \bigcup_{W \in \mathcal{W}^{\mathrm{done}}_g} W : R \cap L(s,t) \neq \emptyset\}| \leq \sum_{i \in \{i', \dots, k\} : n_{g,i} > 1/\varepsilon} n_{g,i}$. Recall that each rectangle R in some row $W \in \mathcal{W}_g$ fulfills $(1+\varepsilon)^p \leq p(R) < (1+\varepsilon)^{p+1}$. This implies

$$\begin{aligned}
p(\{R \in \mathrm{APX}_g \setminus \bigcup_{W \in \mathcal{W}^{\mathrm{done}}_g} W : R \cap L(s,t) \neq \emptyset\}) &\geq (1+\varepsilon)^p \sum_{i \in \{i', \dots, k\} : n_{g,i} > 1/\varepsilon} \lfloor (1+2\varepsilon) n_{g,i} \rfloor \\
&\geq (1+\varepsilon)^{p+1} \sum_{i \in \{i', \dots, k\} : n_{g,i} > 1/\varepsilon} n_{g,i} \\
&\geq p(\{R \in S_g \setminus \bigcup_{W \in \mathcal{W}^{\mathrm{done}}_g} W : R \cap L(s,t) \neq \emptyset\})
\end{aligned}$$

And as $\{R \in \mathrm{APX}_g \cap \bigcup_{W \in \mathcal{W}^{\mathrm{done}}_g} W\} = \{R \in S_g \cap \bigcup_{W \in \mathcal{W}^{\mathrm{done}}_g} W\}$, this completes the proof of the lemma. □

Finally, we define the rays $\mathcal{L}'_{\mathrm{left}}$ and $\mathcal{L}'_{\mathrm{right}}$ for the left and right subproblem, respectively. Intuitively, we define that $\mathcal{L}'_{\mathrm{left}}$ contain all rays in $\mathcal{L}'$ that intersect A_{left}, but we adjust their demands if they intersect some rectangles in APX_g for some groups $g \in \mathcal{G}$. Formally, for each ray $L(s,t) \in \mathcal{L}'$ with $(t,s) \in A_{\mathrm{left}}$ we add a

ray $L'(s,t)$ to $\mathcal{L}'_{\text{left}}$ with a demand of $d(L'(s,t)) := \max\{0, d(L(s,t)) - \sum_{g\in\mathcal{G}} \sum_{R\in \text{APX}_g : R\cap L(s,t)\neq\emptyset} p(R)\}$. We define the rays $\mathcal{L}'_{\text{right}}$ analogously. We recursively compute solutions APX_{left} and $\text{APX}_{\text{right}}$ for the left and right subproblem. If these solutions exist, we obtain the candidate solution $\bigcup_{g\in\mathcal{G}} \text{APX}_g \cup \text{APX}_{\text{left}} \cup \text{APX}_{\text{right}}$. We output the minimal cost candidate solution APX among all guesses. This completes the description of the algorithm.

5.3 Analysis

We show that the computed solution is feasible, has small cost, and that our algorithm has a running time of $2^{(1/\delta\cdot\log C\cdot K\cdot\log(n\cdot p_{\max}\cdot T)/\varepsilon)^{O(K)}}$. First, we show that S restricted to the rectangles in $\mathcal{R}'_{\text{left}}$ is a feasible solution for the left subproblem and S restricted to the rectangles in $\mathcal{R}'_{\text{right}}$ is a feasible solution for the right subproblem, as the following lemma shows.

Lemma 33. *Let $S_{\text{left}} = \mathcal{R}'_{\text{left}} \cap S$ and $S_{\text{right}} = \mathcal{R}'_{\text{right}} \cap S$. Then S_{left} is a feasible solution for the left subproblem and S_{right} is a feasible solution for the right subproblem.*

Proof. We prove that S_{left} is a feasible solution for the left subproblem, the proof that S_{right} is a feasible solution for the right subproblem is analogous to the proof for S_{left}. From the definition of S_{left} and the fact that S contains a prefix in each row, it follows that S_{left} contains a prefix in each row as well. So it suffices to show that S_{left} covers all rays an $\mathcal{L}'_{\text{left}}$. Let $L'(s,t) \in \mathcal{L}'_{\text{left}}$ be a ray. Then there is a ray $L(s,t) \in \mathcal{L}'$ and $d(L'(s,t)) := \max\{0, d(L(s,t)) - \sum_{g\in\mathcal{G}} \sum_{R\in\text{APX}_g : R\cap L(s,t)\neq\emptyset} p(R)\}$. Note that there is no rectangle $R \in \mathcal{R}'_{\text{right}}$ with $R \cap L(s,t) \neq \emptyset$. So

$$\begin{aligned} p(\{R \in S_{\text{left}} : R \cap L(s,t) \neq \emptyset\}) &= p(\{R \in S : R \cap L(s,t) \neq \emptyset\}) - \sum_{g\in\mathcal{G}} p(\{R \in S_g : R \cap L(s,t) \neq \emptyset\}) \\ &\geq d(L(s,t)) - \sum_{g\in\mathcal{G}} p(\{R \in \text{APX}_g : R \cap L(s,t) \neq \emptyset\}) \\ &= d(L'(s,t)) \end{aligned}$$

where the inequality follows from Lemma 32. This show that S_{left} is a feasible solution and thus completes the proof. □

So we actually obtain a solution APX. Now we prove that APX is feasible.

Lemma 34. *The set* APX *is a feasible solution.*

Proof. Suppose that we recursively obtained solutions APX_{left} and $\text{APX}_{\text{right}}$ for the left and right subproblem. Note that by Lemma 33 this is at least the case for the correct guesses. Then APX_{left} and $\text{APX}_{\text{right}}$ contain a prefix of the rectangles in each row, as they are feasible solutions to the respective subproblems. For each $g \in \mathcal{G}$ the set APX_g contains a prefix from each row by construction. Thus, also APX contains a prefix from each row. So it remains to show that every ray $L(s,t) \in \mathcal{L}'$ is covered. Suppose that $L(s,t)$ intersects A_{left}, as the proof for the case when $L(s,t)$ intersects A_{right} is analogue. Then there is a ray $L'(s,t) \in \mathcal{L}'_{\text{left}}$ with $d(L'(s,t)) := \max\{0, d(L(s,t)) - \sum_{g\in\mathcal{G}} \sum_{R\in\text{APX}_g : R\cap L(s,t)\neq\emptyset} p(R)\}$. As APX_{left} is a feasible solution for the left subproblem, we have $p(\{R \in \text{APX}_{\text{left}} : R \cap L(s,t) \neq \emptyset\}) \geq d(L'(s,t))$.

And as $\mathrm{APX}_{\mathrm{left}}$ and the sets APX_g for $g \in \mathcal{G}$ are disjoint, we have

$$\begin{aligned} p(\{R \in \mathrm{APX} : R \cap L(s,t) \neq \emptyset\}) &= p(\{R \in \mathrm{APX}_{\mathrm{left}} : R \cap L(s,t) \neq \emptyset\}) + \sum_{g \in \mathcal{G}} p(\{R \in \mathrm{APX}_g : R \cap L(s,t) \neq \emptyset\}) \\ &\geq d(L'(s,t)) + \sum_{g \in \mathcal{G}} p(\{R \in \mathrm{APX}_g : R \cap L(s,t) \neq \emptyset\}) \\ &\geq d(L(s,t)) \end{aligned}$$

This shows that $L(s,t)$ is covered and thus completes the proof. □

Next, we bound the cost of APX. Lemma 31 already bounded the cost of APX_g for each $g \in \mathcal{G}$. Thus, it remains only to bound the cost of $\mathrm{APX}_{\mathrm{left}}$ and $\mathrm{APX}_{\mathrm{right}}$. By combining Lemmas 31 and 33 we obtain an approximation factor of $1 + 5\varepsilon$.

Lemma 35. *We have that* $c(\mathrm{APX}) \leq (1 + 5\varepsilon)c(S)$.

Proof. We prove the lemma by induction on $\mathrm{left}(A) - \mathrm{right}(A)$. In the base case where $\mathrm{left}(A) - \mathrm{right}(A) = 1/2$, we solve the problem optimally as then $\mathrm{APX} = S_g = \emptyset$. So suppose $\mathrm{left}(A) - \mathrm{right}(A) > 1/2$. Then by induction and Lemma 33, we have that $c(\mathrm{APX}_{\mathrm{left}}) \leq (1 + 5\varepsilon)c(S_{\mathrm{left}})$ and $c(\mathrm{APX}_{\mathrm{right}}) \leq (1 + 5\varepsilon)c(S_{\mathrm{right}})$. By Lemma 31 we have $c(\mathrm{APX}_g) \leq (1 + 5\varepsilon)c(S_g)$. Altogether, we obtain

$$\begin{aligned} c(\mathrm{APX}) &= c(\mathrm{APX}_{\mathrm{left}}) + c(\mathrm{APX}_{\mathrm{right}}) + \sum_{g \in \mathcal{G}} c(\mathrm{APX}_g) \\ &\leq (1 + 5\varepsilon)c(S_{\mathrm{left}}) + (1 + 5\varepsilon)c(S_{\mathrm{right}}) + \sum_{g \in \mathcal{G}} (1 + 5\varepsilon)c(S_g) \\ &= (1 + 5\varepsilon)c(S) \end{aligned}$$

□

Thus it only remains to bound the running time.

Lemma 36. *The running time of the algorithm can be bounded by* $2^{(1/\delta \cdot \log C \cdot K \cdot \log(n \cdot p_{\max} \cdot T)/\varepsilon)^{O(K)}}$.

Proof. The recursion depth of our algorithm is $O(\log T)$. In each recursive call we have $|\mathcal{G}| \leq (1/\delta \cdot \log C \cdot \log p_{\max} \cdot \log T/\varepsilon)^{O(K)}$. For each $g \in \mathcal{G}$ and each $i \in [K]$, we need to guess $n_{g,i}$ and if $n_{g,i} \leq 1/\varepsilon$ we also guess the rows $\mathcal{W}^{\mathrm{cross}}_{g,i}$. As $n_{g,i} \leq n$ and there are at most n rows, this can be done in $n^{O(K/\varepsilon)}$ per group g. For fixed guesses the remaining computation can be done in time $O(n)$. Thus, the total running time is $((n^{O(K/\varepsilon)})^{|\mathcal{G}|})^{O(\log T)}$. By Lemma 30, this can be bounded by $2^{(1/\delta \cdot \log C \cdot K \cdot \log(n \cdot p_{\max} \cdot T)/\varepsilon)^{O(K)}}$. □

Altogether, this implies Lemma 27.

Proof of Lemma 27. By Lemma 34 the algorithm computes a feasible solution to every subproblem, so also to the root subproblem which is the output of the algorithm. By Lemma 35 we have $c(\mathrm{APX}) \leq (1+5\varepsilon)c(S)$, we obtain a $(1+5\varepsilon)$-approximation algorithm. By Lemma 36, the running time is bounded by $2^{(1/\delta \cdot \log C \cdot K \cdot \log(n \cdot p_{\max} \cdot T)/\varepsilon)^{O(K)}}$. So we can apply Lemma 29, which yields the claimed result by rescaling ε. □

References

[1] Foto Afrati, Evripidis Bampis, Chandra Chekuri, David Karger, Claire Kenyon, Sanjeev Khanna, Ioannis Milis, Maurice Queyranne, Martin Skutella, Clifford Stein, et al. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of FOCS*, pages 32–43. IEEE, 1999.

[2] Foto Afrati, Evripidis Bampis, Claire Kenyon, and Ioannis Milis. A ptas for the average weighted completion time problem on unrelated machines. *Journal of scheduling*, 3(6):323–332, 2000.

[3] Antonios Antoniadis, Ruben Hoeksma, Julie Meißner, José Verschae, and Andreas Wiese. A QPTAS for the general scheduling problem with identical release dates. In *Proceedings of ICALP*, volume 80, pages 31:1–31:14, 2017.

[4] Alexander Armbruster, Fabrizio Grandoni, Edin Husić, Antoine Tinguely, and Andreas Wiese. On the approximability of unsplittable flow on a path with time windows. In *Proceedings of IPCO*, volume 15620, pages 29–42, 2025.

[5] Alexander Armbruster, Lars Rohwedder, and Andreas Wiese. A PTAS for minimizing weighted flow time on a single machine. In *Proceedings of STOC*, pages 1335–1344, 2023.

[6] Alexander Armbruster, Lars Rohwedder, and Andreas Wiese. Simpler constant factor approximation algorithms for weighted flow time - now for any p-norm. In *Proceedings of SOSA*, pages 63–81. SIAM, 2024.

[7] Yossi Azar and Noam Touitou. Improved online algorithm for weighted flow time. In *Proceedings of FOCS*, pages 427–437, 2018.

[8] Nikhil Bansal and Jatin Batra. Non-uniform geometric set cover and scheduling on multiple machines. In *Proceedings of SODA*, pages 3011–3021, 2021.

[9] Nikhil Bansal and Ho-Leung Chan. Weighted flow time does not admit o(1)-competitive algorithms. In *Proceedings of SODA*, pages 1238–1244, 2009.

[10] Nikhil Bansal and Kirk Pruhs. The geometry of scheduling. *SIAM Journal on Computing*, 43(5):1684–1698, 2014.

[11] Jatin Batra, Naveen Garg, and Amit Kumar. Constant factor approximation algorithm for weighted flow-time on a single machine in pseudopolynomial time. *SIAM Journal on Computing*, 52(6):FOCS18–158–FOCS18–188, 2020.

[12] Karl Bringmann, Nick Fischer, Danny Hermelin, Dvir Shabtay, and Philip Wellnitz. Faster minimization of tardy processing time on a single machine. In *Proceedings of ICALP*, volume 168, pages 19:1–19:12, 2020.

[13] Chandra Chekuri and Sanjeev Khanna. A PTAS for minimizing weighted completion time on uniformly related machines. In *Proceedings of ICALP*, volume 2076, pages 848–861, 2001.

[14] Chandra Chekuri and Sanjeev Khanna. Approximation schemes for preemptive weighted flow time. In *Proceedings of STOC*, pages 297–305, 2002.

[15] Uriel Feige, Janardhan Kulkarni, and Shi Li. A polynomial time constant approximation for minimizing total weighted flow-time. In *Proceedings of SODA*, pages 1585–1595, 2019.

[16] Nick Fischer and Leo Wennmann. Minimizing tardy processing time on a single machine in near-linear time. In *Proceedings of ICALP*, volume 297, pages 64:1–64:15, 2024.

[17] Kim-Manuel Klein, Adam Polak, and Lars Rohwedder. On minimizing tardy processing time, max-min skewed convolution, and triangular structured ilps. In *Proceedings of SODA*, pages 2947–2960, 2023.

[18] Eugene L Lawler and J Michael Moore. A functional equation and its application to resource allocation and sequencing problems. *Management science*, 16(1):77–84, 1969.

[19] Jan Karel Lenstra and David B Shmoys. Elements of scheduling. *arXiv preprint arXiv:2001.06005*, 2020.

[20] Benjamin Moseley. Scheduling to approximate minimization objectives on identical machines. In *Proceedings of ICALP*, volume 132, pages 86:1–86:14.

[21] Lars Rohwedder and Andreas Wiese. A $(2+\varepsilon)$-approximation algorithm for preemptive weighted flow time on a single machine. In *Proceedings of STOC*, pages 1042–1055, 2021.