# Fair Scheduling via Iterative Quasi-Uniform Sampling

Sungjin Im* Benjamin Moseley†

**Abstract**

In the paper we consider minimizing the $\ell_k$-norms of flow time on a single machine offline using a preemptive scheduler for $k \geq 1$. We show the *first* $O(1)$-approximation for the problem, improving upon the previous best $O(\log \log P)$-approximation by Bansal and Pruhs (FOCS 09 and SICOMP 14) where $P$ is the ratio of the maximum job size to the minimum. Our main technical ingredient is a novel combination of quasi-uniform sampling and iterative rounding, which is of interest in its own right.

## 1 Introduction

Client-server scheduling is a central problem in various fields and there is a wide range of research on the topic [26, 28]. Typically there is a server, or alternatively machine, which receives $n$ requests for jobs to be processed on the machine. Each job $j$ requires $p_j$ time to be processed, arrives at some time $r_j$ and can only begin processing after the job arrives. In some cases, the jobs have priorities. In this case, each job $j$ is associated with a positive weight $w_j$ where a higher weight implies a higher priority. The goal is for the scheduler to process the jobs in order to optimize a quality of service metric for the clients. In this work, we consider scheduling jobs that arrive over time on a single machine that are to be scheduled preemptively offline where the job arrivals and processing times are known in advance.

Given a scheduler, a job is completed at the first time $C_j$ when the scheduler performs $p_j$ units of processing on job $j$. Many objective functions are considered in scheduling theory because different systems have different needs. A widely considered class of objective functions are those that depend on the completion time of jobs, such as total completion time $\sum_i C_i$, total weighted completion time $\sum_i w_i C_i$ and maximum completion time (makespan) $\max_i C_i$. These objectives are well understood in many scheduling environments [16, 14, 1, 2, 23].

While completion time objectives have been widely considered, in the client-server scheduling setting practitioners and theoreticians usually consider scheduling metrics based on the flow time of the jobs. The flow time of $j$ is $C_j - r_j$, which is the time the job $j$ waits to be completed since its arrival. When jobs arrive over time, it is of more interest to consider the flow time of a job rather than the completion time. This is because the completion time ignores when the jobs arrive, which is critical for capturing the quality of service a client receives. In particular, a client that submits jobs $j$ would like $j$ to be completed as soon as possible. In other words, the client would like the flow time of $j$ to be minimized. In the client-server setting, the scheduler typically considers a quality of service metric over all the jobs' flow times.

One of the most popular objectives is minimizing the total (or equivalently average) flow time $\sum_j (C_j - r_j)$. This objective focuses on optimizing the average quality of service of the jobs or, alternatively, the average waiting time of the system. In the case where the jobs have priorities, the goal is to find a scheduler minimizing the total weighted flow time $\sum_j w_j (C_j - r_j)$. While total (weighted) flow time is a fundamental objective, unfortunately an algorithm that minimizes the total flow time of a schedule may be unfair to individual jobs. Amongst other priorities, fairness is one of the highest concerns in almost all systems in practice [29]. Due to unfairness, it is not surprising that many systems in practice do not use algorithms that are designed to minimize total flow time only.

To enforce fairness into the schedule, the most commonly considered metric is minimizing the $\ell_k$-norm of the flow times $\sqrt[k]{\sum_j (C_j - r_j)^k}$. For the $\ell_k$-norms, typically $k \in [2, 3]$, in which case the schedule is optimizing the variance of flow times of the jobs. By optimizing the variance, the scheduler enforces fairness among the clients. Due to the importance of fairness criteria, the $\ell_k$-norms of flow time has been extensively studied in many machine environments in search of discovering the most efficient schedulers. For example: on a single machine [6, 7], multiple machines [12, 10, 8, 19, 24], in broadcast scheduling [17, 13, 21], for parallel processors [17, 21] and on speed scalable processors [22].

The setting we consider, preemptive job scheduling

*Electrical Engineering and Computer Science, University of California, 5200 N. Lake Road, Merced CA 95344. sim3@ucmerced.edu. Supported in part by NSF grants CCF-1409130 and CCF-1617653.

†Department of Computer Science and Engineering, Washington University in St. Louis, 1 Brookings Drive, St. Louis, MO 63130. bmoseley@wustl.edu. Supported in part by a Google Research Award, a Yahoo Research Award and NSF Grant CCF-1617724.

on a single machine, has been studied for decades. However, despite being well researched, the complexity of some of the the most fundamental quality of service objectives are not known. This is despite the fact that this is perhaps the most basic environment considered in the client-server setting. In particular, the complexity of the total weighted flow and the $\ell_k$-norms of flow time remain unresolved. The likely reason is that flow time based objectives require fundamentally different, and in many cases more sophisticated, algorithmic techniques than completion time objectives.

Two central open problems in the client-server scheduling setting are if minimizing the total weighted flow time and/or the $\ell_k$-norms of flow time admit an $O(1)$-approximation. It is known that unweighted total flow time can be solved optimally in polynomial time using the shortest-remaining-processing-time algorithm. Weighted flow time was shown to be strongly NP-Hard [25] over three decades ago. The $\ell_k$-norms of flow time for $k \leq 2 < \infty$ had resisted a hardness proof until recently it was shown to be strongly NP-Hard for all $k \leq 2 < \infty$ [27]. Neither problem is known to be APX-Hard.

The best known algorithms for both problems were shown in [7] which gave a $O(\log \log P)$ [1] approximation algorithm in both cases where $P$ is the ratio of the maximum to minimum job size. If all jobs arrive at the same time, the shortest job first algorithm is optimal for the $\ell_k$-norms of flow time and Smith's rule is an optimal algorithm for weighted flow time. A quasi-polynomial time approximation scheme is known for weighted flow time [15]. Due to the existence of this quasi-polynomial time algorithm, it is reasonable to believe that total weighted flow time admits an $O(1)$-approximation. As will be discussed shortly, the $\ell_k$-norms of flow time is similar to weighted flow time, so it is additionally reasonable to believe this problem admits an $O(1)$-approximation.

Similar techniques have been used to develop algorithms for both problems [18, 5, 4, 3, 20]. This is natural as the objectives behave similarly mathematically. Indeed, consider the total weighted flow time objective and some fixed schedule $A$. Let $W^A(t)$ denote the total weight of the jobs that have arrived but are unsatisfied in $A$ at time $t$. Then the objective value of $A$ is $\int_{t=1}^{\infty} W^A(t)$. Thus the goal is to ensure the algorithm $A$ has small aggregate weight of jobs that are unsatisfied at each time step on average in the schedule. Similarly, for the $\ell_2$-norm of flow, let $L^A(t)$ denote the sum of the *ages* of the jobs that have arrived and are unsatisfied in a schedule $A$ at time $t$. The age of a job $j$ at time $t$ is $t - r_j$. The $\ell_2$-norm objective of $A$, ignoring the outer square root, is $\int_{t=1}^{\infty} 2L^A(t)$. Here the goal is to ensure the algorithm has small aggregate age of jobs that are unsatisfied at each time step on average in the schedule.

The similarity in the objectives is that the ages act very much like weights. Due to this, algorithms for one objective tend to lead to the discovery of an algorithm for the other objective. Even the NP-Hardness reduction for the $\ell_k$-norm of flow time essentially releases jobs in a very careful way to ensure that they simulate having weights, effectively simulating the reduction used to show weighted flow is NP-Hard.

While both objectives are similar, they are obviously not the same. In particular, the ages of jobs change over time, which can make them more challenging algorithmically than weights. Alternatively, the weights for jobs can be arbitrary and need not depend on when jobs arrive, whereas the ages of jobs depend on the jobs' arrival time. This fact makes the ages of the jobs less algorithmically challenging than weights. Due to this, the problems are mathematically incomparable.

In the research that has tried to answer whether or not these two problems admit $O(1)$-approximation algorithms, one pervasive question is whether or not one of these problems is easier than the other.

**Results:** In this paper we consider the $\ell_k$-norms of flow time objective on a single machine. We show the following theorem.

THEOREM 1.1. *There is a randomized polynomial time $O(1)$-approximation algorithm for the $\ell_k$-norm of flow time for all $k \geq 1$ in the single machine setting.*

To show the previous theorem, we strongly use a key property of the $\ell_k$-norms of flow time, which does not hold in the case of average weighted flow time even for the $\ell_1$ norm. The property is the following: for any two jobs $i$ and $j$ such that $r_i \leq r_j$ and $p_i < p_j$, one can assume w.l.o.g. that $i$ is completed before $j$ in an optimal schedule. This is because if $j$ completes earlier, then one could have finished $i$ by $j$'s completion time since it has smaller processing time. Further, since $i$ arrives earlier, it only costs the scheduler more for the $\ell_k$-norms objective by making $i$ wait longer than $j$ to be satisfied ($i$'s age is more than $j$'s at any time). The proof of this follows by this intuition and an elementary swapping argument. While this property is simple, it is the key underlying property which allows us to apply our algorithm and analysis framework.

We note that this property does not hold in the weighted case even for the $\ell_1$-norm. In particular, if $r_i \leq r_j$ and $p_i < p_j$ one may need to complete $j$ before $i$ if $j$'s weight is much larger than $i$'s weight. Perhaps this property makes the $\ell_k$-norms of flow time an easier problem than weighted flow time.

[1] More precisely, one can get a $O((\log \log P)^{1/k})$-approximation for the $\ell_k$-norm of flow using the algorithm in [7] since it gives a $O(\log \log P)$-approximation for the $k$th power of flow, i.e. $\sum_j (C_j - r_j)^k$.

**1.1 An Overview of the Analysis Techniques** We now discuss the core techniques we use at a high level. The most relevant work to ours is that of [7]. The work of [7] reduces the $\ell_k$ norm problem to a geometric set cover problem so that their algorithm can leverage known techniques for geometric set cover. Unlike this previous work, we will consider the $\ell_k$-norm problem directly.

We begin by using an integer program for the $\ell_k$-norms problem, which was introduced in [7]. The integer program has a variable $x_{i,t}$ that is 1 if job $i$ is satisfied at time $t$ and 0 otherwise. The last time $t$ where $x_{i,t} = 1$ is the time job $i$ is completed. Effectively, this sets 'deadlines' for the jobs. The constraints of the program ensure that a feasible schedule can finish the jobs by their deadlines by enforcing that the amount of work that must be done during every time interval does not exceed the length of the interval. Our algorithm relaxes the program to a linear program and solves this LP. Then the algorithm rounds it to an integral solution whose expected cost is bounded by a $O(1)$ factor of the optimal LP cost.

To round the LP, we need to set integral completion times of jobs and ensure all the constraints of the LP are satisfied. To do this, our algorithm samples completion times for jobs via randomized rounding, oversampling a completion time by an $O(1)$ factor more than the fractional LP value. By oversampling by an $O(1)$ factor, we ensure the completion times sampled only cost $O(1)$ more than the optimal LP cost. Unfortunately, this may leave some constraints unsatisfied. In particular, there could be no way for a scheduler to meet the sampled completion times for all the jobs. The algorithm needs to push back the completion times of some jobs to later times to obtain a feasible schedule.

At this point, we could recurse and use iterative rounding to ensure the remaining constraints are satisfied. However, we cannot recurse too many times using standard iterative rounding while keeping the cost to be at most a $O(1)$ factor larger than the LP's cost. In the end, the algorithm needs to ensure each job's completion time is only oversampled by an $O(1)$ factor to be able to bound the objective.

To circumvent this hurdle our algorithm defines for each unsatisfied constraint, corresponding to an overloaded time interval $I$, a set of jobs $N_I$ that are critical for satisfying this constraint. This set may not be all jobs being used to satisfy the constraint for $I$ in the LP, but only the critical ones. The set is defined in such a way to take advantage of the property of optimal solutions to the $\ell_k$-norm problem mentioned above. Intuitively, jobs in $N_I$ are the most important jobs the LP used to satisfy the constraint for $I$. Our algorithm does recurse and utilize iterative rounding; however, the algorithm only considers jobs that are in $N_I$ for some interval $I$ whose constraint is not satisfied. We show that the probability that a fixed job is included in *any* set $N_I$ over all intervals $I$ is a small constant. Thus, in each iteration, the probability a completion time for a job is sampled decreases geometrically because the probability a job remains in the LP decreases geometrically. [2] This ensures that overall any completion time is still sampled with $O(1)$ extra boosting over the fractional amount in the LP, as if only one iteration of randomized rounding occurred. For this reason, we call our approach *iterative quasi-uniform sampling*. After recursing $O(\log n)$ times, we will be able to show all intervals are satisfied, even though we only considered jobs in $N_I$ to satisfy the constraint for $I$.

Our technique is in similar spirit as the quasi-uniform sampling technique of [30, 11] for geometric set cover; however, this previous work does not consider LP rounding and is quite different.

## 2 Preliminaries

There is a set of $n$ jobs that arrive over time. Each job $j$ has a processing time/size $p_j$ and arrives at some time $r_j \geq 0$. Both quantities $p_j \geq 1$ and $r_j$ are assumed to be integers. Times are slotted. During each time slot $[t, t+1]$ for an integer $t \geq 0$, one can either process only one job exactly by one unit or do nothing. The goal is to schedule all jobs preemptively offline to minimize the $\ell_k$ norm of flow time. A job $j$ completes at the earliest time when it is processed by $p_j$ units since its arrival. If job $j$ is completed at time $C_j$, the objective is $(\sum_j (C_j - r_j)^k)^{1/k}$; each individual job $j$'s flow time is $C_j - r_j$. Note that each job has flow time at least 1. Throughout the paper, we will focus on minimizing the $k$th power of flow time $\sum_j (C_j - r_j)^k$ ignoring the outer $k$th root. Note that if one finds a schedule that is $O(1)^k$-approximate for the $k$th power of flow time, then it is an $O(1)$-approximation for the $k$th norm of flow time. For an interval $I = [s, t]$ with integers $0 \leq s \leq t$, $|I|$ is defined as $t - s$, i.e., the number of time slots in the interval. We let $a(I)$ and $b(I)$ denote the interval $I$'s starting and ending times, respectively. So, $a(I) = s$ and $b(I) = t$ if $I = [s, t]$.

We begin by giving a linear programming relaxation, $\mathsf{LP}_{\mathsf{main}}$ for the $k$th power of flow time objective, which was introduced in [7]. To make our presentation more transparent, we assume that each job's processing time is polynomially bounded by $n$. This assumption allows us to ensure the number of time steps considered in the linear programming is polynomially bounded by $n$. This simplifying assumption will be removed in Appendix C. For completeness, we briefly discuss why $\mathsf{LP}_{\mathsf{main}}$ is a

[2] More precisely, if a job $j$ is almost complete at time $t$, then we show that $j$'s completion time can be safely upper bounded by time $t$ with a constant probability in each iteration, making the LP solution more integral by decreasing the LP cost due to fractional $x_{j,t}$ by a constant factor.

valid relaxation. To see this, restrict the values of $x_{j,t}$ to 0 or 1. Then, the variable $x_{j,t}$ becomes an indicator variable that is one if and only if $j$ is alive at time $t$ (more precisely, during time slot $[t-1,t]$). In other words, the latest time $t$ where $x_{j,t} = 1$ after $r_j$ is the completion time of $j$. The objective follows since a job $j$ alive at time $t \geq r_j + 1$ increases its $k$th power of flow time by $\Delta_{t-r_j} := (t-r_j)^k - (t-r_j-1)^k$ during time slot $[t-1,t]$. Let $P(S) = \sum_{j \in S} p_j$ denote the total processing time of jobs in a set $S$. Let $R(I)$ denote the set of jobs that arrive during $I = [t_1, t_2]$. For a time interval $I$, define $V(I) := P(R(I)) = \sum_{j, r_j \in I} p_j$ as the total size/volume of jobs arriving during $I$.

We only discuss Constraints (2.2) since other constraints are obvious. To gain an intuition, consider a simpler constraint $\sum_{j \in R(I)} p_j x_{j,t} \geq V(I) - |I|$ by setting $S = \emptyset$ and ignoring the min. Then, for any time interval $I = [s,t]$, at most $|I|$ units of work can be processed. Hence the total size of jobs that arrive during $I$ and are still alive at the end of $I$ (the left-hand-side) must be no smaller than the total size of jobs arriving during $I$ minus $|I|$, (the right-hand-side). Constraints (2.2) are standard knapsack covering inequalities [9] for this covering constraint and requires the demand $V(I) - |I|$ to be covered by jobs not in $S$ by at least $V(I) - |I| - P(S)$, which is clearly true for all integer solutions. The min truncates each job's size from $p_j$ to $V(I) - |I| - P(S)$, and it has no effect on the feasibility of integer solutions. We obtain the LP relaxation by extending $x_{j,t} \in \{0,1\}$ to $x_{j,t} \geq 0$. So for each interval $I$, we have a collection of knapsack covering inequalities. The LP can be solved using the Ellipsoid method to arbitrarily close to the optimum; only the objective achieved is approximate and all constraints are satisfied. For more details, see [9].

Our algorithm solves $\mathsf{LP}_{\mathsf{main}}$ and rounds the fractional solution to an integral solution. Once we have an integral solution, it is easy to transform this into a feasible assignment. In particular, once jobs have integral completion times, one can treat these completion times as deadlines. By using the Earliest-Deadline-First (EDF) algorithm we can ensure all jobs are completed by their deadline. The proof is not difficult and is deferred to Appendix D.

LEMMA 2.1. *For any feasible integral solution $x$ to $\mathsf{LP}_{\mathsf{main}}$, there is a polynomial time algorithm that constructs a schedule of cost no more than the LP cost of the LP solution $x$.*

Given the previous lemma, our goal is to round a feasible solution to $\mathsf{LP}_{\mathsf{main}}$ to a feasible integral solution. If we can show that the integral solution has cost at most an $O(1)^k$ factor larger than $\mathsf{LP}_{\mathsf{main}}$ then this will complete the proof of our main theorem. The goal of our algorithm is to discover a completion time $C_j^*$ for each job $j$ such that the cost of these completion times for the jobs is not too large. Our algorithm works by using several procedures to set the completion time of the jobs. In particular, in each step, we will set the completion time of some jobs $j$ to be at least some time. Then at the end we set $C_j^*$ to be the maximum of all lower bounds to job $j$'s completion time we set in all steps. This does not affect the feasibility of the LP solution since increasing a job's completion time never decreases $x_{j,t}$ values, thus guaranteeing Constraints (2.2) remain satisfied.

The algorithm begins by using threshold rounding. Let $c \leq \frac{1}{16 \cdot 2^k}$ be a constant. Let $C_{j,c}$ be the latest time $t$ where $x_{j,t} \geq c$. For every job $j$, we set the completion time of $j$ to be at least $C_{j,c}$. Note that this ensures that $x_{j,t'} = 1$ for all $r_j \leq t' \leq C_{j,c}$, regardless of the remaining rounding steps. We note that the increase of the objective due to this rounding can easily be bounded by the cost of the LP. We let $\mathrm{OPT}(\mathrm{LP})$ denote the optimum for the LP. The proof of the following claim is straightforward and is deferred to Appendix D.

CLAIM 2.2. *After the threshold rounding, the LP cost is at most $\frac{1}{c} \cdot \mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$.*

For each interval $I$, let $S_{c,I}$ be the set of jobs that arrive during $I$ that are given a deadline after the end of the interval $I$ by the threshold rounding, i.e. $S_{c,I} := \{j \mid r_j \in I \text{ and } C_{j,c} \geq b(I)\}$. Note that this threshold rounding may satisfy all constraints (2.2) for some intervals $I$. In particular, for any interval $I$ where $P(S_{c,I}) \geq V(I) - |I|$.

Consider the remaining set of intervals. Let $H_I$ be the set of jobs in $R(I) \setminus S_{c,I}$ where $p_j \geq V(I) - |I| - P(S_{c,I})$ and let $L_I$ be the remaining jobs in $R(I) \setminus S_{c,I}$. We say that an interval $I = [t_1, t_2]$ that is not satisfied by the threshold rounding is *heavy* if $\sum_{j \in H_I} x_{j,t_2} \geq \frac{1}{2}$ and the interval is *light* otherwise, i.e. $\sum_{j \in L_I} p_j x_{j,t_2} \geq \frac{1}{2}(V(I) - |I| - P(S_{c,I}))$. Let $\mathcal{H}$ be the set of heavy intervals and $\mathcal{L}$ be the set of light intervals. In the remaining sections, we set the completion times of jobs separately to satisfy light intervals and heavy intervals. By taking the maximum completion time from each case for a job, we can ensure all constraints corresponding to intervals are satisfied. The main technical challenge arises in the heavy case. The proof for the light case is similar to [7] and, for completeness, the proof can be found in Appendix B.

To round both the heavy and light cases, we will bound the cost of the completion times chosen by at most a $\frac{1}{c} O(1)^k$ factor larger than the optimum LP cost. This will show that our algorithm is an $\frac{1}{c} O(1)^k$ approximation for minimizing the $k$th power of flow time. By taking the outer $k$th root for the $\ell_k$ norm objective, we obtain an $O(1)$ approximation for the $\ell_k$ norm by fixing $c$ to be $\frac{1}{2^{k+4}}$.

$$(\mathsf{LP}_{\mathsf{main}}) \qquad \min \quad \sum_j \sum_{t>r_j} \Big((t-r_j)^k - (t-1-r_j)^k\Big) x_{j,t}$$

$$\text{(2.1)} \qquad \text{s.t.} \qquad x_{j,t} \le x_{j,t-1} \qquad \forall j, t > r_j$$

$$\sum_{j \in R(I)\setminus S} x_{j,t} \min\{p_j, V(I) - |I| - P(S)\} \ge V(I) - |I| - P(S)$$

$$\text{(2.2)} \qquad \forall I = [s,t], \forall S \subseteq R(I) \text{ where } V(I) - |I| - P(S) \ge 0$$

$$x_{j,r_j} = 1 \qquad \forall j$$

$$x_{j,t} \ge 0 \qquad \forall j, t \ge r_j$$

## 3 Heavy Intervals

In this section, our goal is to set the completion times of jobs so that Constraints (2.2) for all intervals in $\mathcal{H}$ are satisfied. To do this, we solve a second linear program, which is more relaxed than the original LP, $\mathsf{LP}_{\mathsf{main}}$. Hence the new LP, $\mathsf{LP}_{\mathsf{heavy}}$ will have an optimal solution cheaper than $\mathsf{LP}_{\mathsf{main}}$. Further, the LP will be strong enough to give a solution that satisfies all constraints corresponding to heavy intervals. We use a more relaxed LP because we will iteratively construct feasible solutions to this LP and having a simpler LP makes this easier. Here we rename variables to distinguish this new LP from $\mathsf{LP}_{\mathsf{main}}$, but they have the same interpretation: $y_{j,t} = 1$ if job $j$ is alive during time slot $[t-1, t]$ and 0 otherwise. Recall that $\Delta_{t-r_j} = (t-r_j)^k - (t-r_j-1)^k$.

$$(\mathsf{LP}_{\mathsf{heavy}}) \qquad \min \sum_j \sum_{t>r_j} \Delta_{t-r_j} y_{j,t}$$

$$\text{(3.3)} \qquad \text{s.t.} \quad y_{j,t} \le y_{j,t-1} \qquad \forall j, t > r_j$$

$$\text{(3.4)} \qquad \sum_{j \in H_I} y_{j,t} \ge 1 \qquad \forall I = [s,t] \in \mathcal{H}$$

$$y_{j,r_j} = 1 \qquad \forall j$$

$$y_{j,t} \ge 0 \qquad \forall j, t \ge r_j$$

$\mathsf{LP}_{\mathsf{heavy}}$ is a relaxation of $\mathsf{LP}_{\mathsf{main}}$. There are several changes. First we simplify Constraints (2.2) to only consider the heavy intervals; as mentioned, the constraints regarding light intervals will be taken care of in Appendix B. Further, to satisfy the constraint for a heavy interval $I$, we will only use jobs in $H_I$. Recall that every job $j$ in $H_I$ has a size no smaller than the residual demand, $V(I) - |I| - P(S_{c,I})$. In other words, we satisfy Constraint (2.2) regarding a heavy interval $I = [s,t]$ if and only if the completion time of at least one job in $H_I$ is set to be at least $t$. Hence we have Constraints (3.4).

CLAIM 3.1. $\mathrm{OPT}(\mathsf{LP}_{\mathsf{heavy}}) \le \frac{2}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$.

*Proof.* Consider the fractional solution $\{x_{j,t}\}$ we obtained in Section 2 after applying the threshold rounding. As observed in Claim 2.2, the solution has cost at most $\frac{1}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$. Set $y_{j,t} = x_{j,t}$. Note that the objective of $\mathsf{LP}_{\mathsf{heavy}}$ for $y$ is exactly the same as that of $\mathsf{LP}_{\mathsf{main}}$ for $x$. Then consider setting $y_{j,t}$ to be $2x_{j,t}$ for all $t > C_{j,c}$ for all jobs $j$, which increases the LP cost by a factor of at most two. This is well-defined since for such times $t$, $x_{j,t} < c \le 1/2$. This satisfies all constraints in the $\mathsf{LP}_{\mathsf{heavy}}$ by definition of heavy intervals.

We now give a crucial definition of a property that is true for any optimal solution to $\mathsf{LP}_{\mathsf{heavy}}$. This definition captures a key structural property of the $\ell_k$ norm of flow time.

DEFINITION 3.2. *We say that a solution $y$ to $\mathsf{LP}_{\mathsf{heavy}}$ is priority-preserving if the following property is satisfied: for any two jobs $j$ and $i$ such that $r_j \le r_i$ and $p_j < p_i$, and any time $t \ge r_i$, if $y_{j,t} > 0$, then $y_{i,t} = 1$. (In other words, if $j$ is still alive at time $t$, then $i$ has not been fractionally completed at all at time $t$.)*

We are now ready to introduce our key lemma, but, before we do, we need to define several notions. For a solution $y'$ to $\mathsf{LP}_{\mathsf{heavy}}$, define $\mathsf{LP}_{\mathsf{heavy}}(y')$ to be $\mathsf{LP}_{\mathsf{heavy}}$'s objective for $y'$. We decompose the cost into two parts, $\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y') := \sum_j \sum_{t>r_j, y'_{j,t}=1} \Delta_{t-r_j}$ and $\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y') := \sum_j \sum_{t>r_j, y'_{j,t}<1} \Delta_{t-r_j} y'_{j,t}$. In words, $\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y')$ [$\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y')$, resp.] denotes the total contribution to the LP's objective from jobs on time steps where they are alive integrally [fractionally, resp.]. Clearly, $\mathsf{LP}_{\mathsf{heavy}}(y') = \mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y') + \mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y')$.

We will show the following key lemma.

LEMMA 3.3. *There exists a randomized polynomial time algorithm which takes as input a feasible priority-preserving solution $y$ to $\mathsf{LP}_{\mathsf{heavy}}$ and constructs a new feasible priority-preserving solution $y^*$ to $\mathsf{LP}_{\mathsf{heavy}}$ such that*

1. $\mathrm{E}[\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y^*)] - \mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y) \le \frac{2^{3k+3}}{c}\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y) = O(1)^k \mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y)$.

2. $\mathrm{E}[\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y^*)] \leq 2^{2k+3}\exp(-(\frac{2}{5c}))\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y) \leq \frac{1}{2}\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y)$.

By repeatedly applying Lemma 3.3, we will convert a fractional solution to $\mathsf{LP}_{\mathsf{heavy}}$ to a integral solution without increasing the cost too much. Property (1) upper bounds the increase of the integral LP cost by $O(1)^k$ times the *fractional* LP cost of the current solution. Property (2) states the fractional LP cost decreases by a constant factor in each iteration, thus making the LP solution more integral. To show these properties, we ensure the invariant that the LP solution always remains priority-preserving and crucially use it for our analysis.

To apply Lemma 3.3 we first need to have an initial solution to $\mathsf{LP}_{\mathsf{heavy}}$ that is priority-preserving, which is captured in the following lemma.

LEMMA 3.4. *There is a feasible priority-preserving solution $y$ to $\mathsf{LP}_{\mathsf{heavy}}$ with cost at most $\frac{2}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$.*

*Proof.* We show that there is an optimal solution that is priority-preserving and has cost at most $\frac{2}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$. For the sake of contradiction, suppose that this claim is false. Consider an optimal solution $y$ to $\mathsf{LP}_{\mathsf{heavy}}$. Let $\tau$ be the *first* time the claim is false for some two jobs $i$ and $j$ where $r_j \leq r_i$ and $p_j < p_i$. This implies that $y_{j,\tau} > 0$, but $y_{i,\tau} < 1$ and $y_{i,\tau-1} = 1$. Let $T'$ be the set of times $t \geq \tau$ such that $y_{j,t} > 0$. Let $\alpha = \min_{t\in T'} \min\{y_{j,t}, 1 - y_{i,t}\}$. For each time $t \in T'$, we decrease $y_{j,t}$ by $\alpha$ and increase $y_{i,t}$ by $\alpha$. Note that this can only reduce the LP objective since we know that $r_j \leq r_i$, meaning that $\Delta_{t-r_j} \geq \Delta_{t-r_i}$. If $r_j < r_i$, then clearly this operation strictly reduces the LP objective. Otherwise, by changing $\Delta_{j,t}$ infinitesimally, we can still ensure the LP objective strictly decreases. [3] Hence we only need to show that the new LP solution $y$ remains feasible. We only show that Constraints (3.4) remain satisfied since it is straightforward to see other constraints are satisfied. To see this, consider any time $t \in T'$. Consider any interval $I$ such that $j \in H_I$ and $b(I) = t \in T'$. Since we decreased $y_{j,t}$ by $\alpha$, the reader may wonder if Constraint (3.4) is violated. However, note that $j \in H_I$ implies that $a(I) \leq r_j \leq b(I) = t$. Since $r_j \leq r_i$, we have $a(I) \leq r_i$. Further, since $r_i \leq \tau \leq t = b(I)$, we have that $a(I) \leq r_i \leq b(I)$. Knowing that $p_i > p_j$ and $p_j \geq V(I) - |I| - P(S_{c,I})$ since $j \in H_I$, we conclude that $i \in H_I$. Hence the decrease of $y_{j,t}$ value can be offset by the increase of $y_{i,t}$. This is a contradiction to the fact that $y$ is an optimal solution to $\mathsf{LP}_{\mathsf{heavy}}$.

[3] For example, we can use the following procedure. Let $M$ be the total number of time steps we need to consider. Rank jobs based on their sizes breaking ties arbitrarily: the smallest job has rank 1 and the largest job has rank $n$. Then, we decrease each $\Delta_{j,t}$, $t > r_j$ by $1/(nM)^{2k}$ times job $j$'s rank.

Thus, we have shown that there is an optimal LP solution $y$ to $\mathsf{LP}_{\mathsf{heavy}}$ that is priority-preserving, and the cost must be at most $\frac{2}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$ by Claim 3.1.

Using Lemma 3.3 and the pervious lemma, we can find an integral solution to $\mathsf{LP}_{\mathsf{heavy}}$ with low cost.

COROLLARY 3.1. *There exists a polynomial time algorithm that produces a feasible* integral *solution to $\mathsf{LP}_{\mathsf{heavy}}$ whose expected cost is at most $\frac{1}{c}\cdot O(1)^k\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$.*

*Proof.* We first show that we only need to apply Lemma 3.3 a logarithmic number of times to arrive at an integral solution to $\mathsf{LP}_{\mathsf{heavy}}$ with high probability. To do this, note that by Property (2) the cost of the fractional portion of the LP decreases by a constant factor each iteration. Thus, we need only show that the procedure terminates once the objective value is less than $1/\mathtt{poly}(n)$. Consider any interval $I \in \mathcal{H}$ that is not satisfied by the current LP solution, $y$. Then there must be a job $j \in H_I$ such that $1 > y_{j,b(I)} \geq 1/n$ due to Constraint (3.4) and the fact that $|H_I| \leq n$. Hence job $j$ contributes to the fractional LP cost by at least $(1/n)1^k = 1/n$. What this implies is that if the objective is smaller than $1/n$, then every interval must be satisfied. Thus, it is easy to see the number of iterations we need to apply Lemma 3.3 is polynomially bounded by the input size with high probability.

Finally, note that the increase of the integral cost of the LP solution in each iteration is upper bounded by $\frac{1}{c}\cdot O(1)^k$ times the fractional cost of the current LP solution (Property (1)), and the fractional cost decreases by a factor of more than $1/2$ in each iteration (Property (2)) since $c = \frac{1}{2^{k+4}}$. Hence by Lemma 3.4, we obtain the claimed upper bound on the expected LP cost.

**3.1 Iterative Quasi-Uniform Sampling (Proof of Lemma 3.3)** Our goal is to construct a new LP solution $y^*$ that satisfies the properties claimed in Lemma 3.3. To prove Lemma 3.3, fix a feasible priority-preserving solution $y$ to $\mathsf{LP}_{\mathsf{heavy}}$, which we can assume wlog due to Lemma 3.4. After each iteration of the randomized rounding, we will have to make the current fractional solution priority-preserving, which is non-trivial. To keep the flow of presentation, we defer this transformation to Appendix A. First, we set $y^*_{j,t} = 1$ if $y_{j,t} = 1$. Next, we use a randomized rounding which consists of the following two steps.

**Step (i).** For each job $j$, define $r'_j$ be the latest time when $y_{j,t} = 1$. For each job $j$ and $i \geq 0$, define $\beta_{j,i} := 2^i + r'_j$; also define $\beta_{j,-1} = r'_j$. For each job $j$ and $i \geq 2$, we sample the completion time $\beta_{j,i}$ independently with probability $\min\{1, \frac{1}{c}\cdot y_{j,\beta_{j,i-2}}\}$ – this sampling is done independent of sampling of other $\beta_{j,i'}$, $i' \neq i$ as well as other jobs. Note that the probability that time

$\beta_{j,i}$ is sampled is $\frac{1}{c}$ multiplied by the amount job $j$ is fractionally unsatisfied at the *earlier* time $\beta_{j,i-2}$. When $\beta_{j,i}$ is sampled, we make $j$'s completion time to be at least $\beta_{j,i}$ by setting $y^*_{j,t} = 1$ for all $r_j \le t \le \beta_{j,i}$. Note that if multiple deadlines are sampled for a job, job $j$'s completion is set to be at least the latest one sampled.

**Step (ii).** After step (i), some interval constraints (Constraints (3.4)) may be already satisfied. In particular, an interval $I = [a(I), b(I)] \in \mathcal{H}$ is satisfied if any job $j$ in $H_I$ is given a completion time no earlier than the ending time of $I$ since then $y^*_{j,b(I)} = 1$. For notational convenience, we remove such intervals from $\mathcal{H}$ and focus on the remaining intervals in $\mathcal{H}$. Note that for any interval $I \in \mathcal{H}$, we have $y_{j,b(I)} \le c$ for all $j \in H_I$. We define a set $N_I$ of jobs for each interval $I \in \mathcal{H}$, which we call *critical* jobs for interval $I$. The set $N_I$ for an interval $I$ in $\mathcal{H}$ is defined by removing the set $E_I$ of jobs in $H_I$ with earliest arrival times until $\sum_{j \in H_I \setminus E_I} y_{j,b(I)} \le \frac{3}{5}$. For each interval $I \in \mathcal{H}$ (whose constraint is not satisfied by step (i)) and for every job $j \in N_I$, we set $y^*_{j,b(I)} = \max\{y^*_{j,b(I)}, 2y_{j,b(I)}\}$. [4] Also to ensure that Constraints (3.3) is satisfied, we increase $y^*_{j,t}$ if necessary for all $r_j \le t \le b(I)$ so that $y^*_{j,t} \ge 2y_{j,b(I)}$.

**Intuition.** Our iterative randomized rounding makes the solution more integral in each iteration. Here the procedure keeps the current solution feasible while decreasing the cost of fractional part of the LP solution by a constant factor in each iteration. In Step (i), we sample a completion time for each job. For ease of analysis, we will only distinguish completion times only when they differ by more than a constant factor in length. This is why we sample $j$'s completion time to be one of the $\beta_{j,i}$ times for some $i$. By using the threshold rounding and 'boosted-up' randomized rounding, we 'oversample' the completion time by up to a constant factor. Sampling $j$'s completion time based on how much the job is alive in the current solution at an earlier time $\beta_{j,i-2}$ has a similar spirit. The LP's integral cost increases by $O(1)^k$ factor of its fractional cost.

Step (ii) is concerned with intervals that are not satisfied by Step (i), and is more interesting. Recall that our goal is to decrease the fractional cost of the solution by a constant factor. The cost is measured by when jobs complete while each constraint is defined over an interval which can be satisfied when jobs arriving during the interval complete later than the end of the interval. To safely remove a job from the current solution (more precisely, fractional completion times for a job), we should be able to find other jobs that still satisfy the intervals (constraints) that the job was used to satisfy. This is why we define critical jobs for each interval. Critical jobs are sufficient to satisfy intervals if their contributions to the constraints are increased by a constant factor. Using the fact that a job is used for intervals only when the job is critical for them, we show that a job is needed after Step (i) with a small probability, which will be more than enough to cancel the effect of increasing critical jobs contributions.

We begin the analysis by showing $y^*$ is feasible.

PROPOSITION 3.1. *For all $I = [a(I), b(I)] \in \mathcal{H}$, we have $\sum_{j \in N_I} y_{j,b(I)} \ge \frac{1}{2}$.*

*Proof.* We use the fact that for any job $j$ in $H_I$, $y_{j,b(I)} \le c \le \frac{1}{10}$ which follows from the definition of $c$ and the threshold rounding.

LEMMA 3.5. *$y^*$ is feasible to $\mathsf{LP}_{\mathsf{heavy}}$.*

*Proof.* It is easy to see that $y^*$ satisfies Constrains (3.3) after both steps (i) and (ii). We only need to show that if Constraints (3.4) for an interval $I$ is not satisfied by step (i), then it is fractionally satisfied by jobs in $N_I$. By Proposition 3.1, we have $\sum_{j \in N_I} y_{j,b(I)} \ge \frac{1}{2}$. Our algorithm sets $y^*_{j,b(I)} \ge 2y_{j,b(I)}$, thus satisfies Constraint (3.4) for $I$.

Next we upper bound the cost of the new LP solution $y^*$ we constructed following steps (i) and (ii). First we bound the cost for times set integrally by step (i). One tricky issue is that $y^*$ as defined may not be priority-preserving. We will modify $y^*$ later to make it priority-preserving. We will first upper bound $\mathrm{E}[\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y^*)] - \mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y)$ and $\mathrm{E}[\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y^*)]$, and will show that the modification increases neither of the costs.

The following inequality is useful for our analysis and follows from an elementary algebra.

PROPOSITION 3.2. *For all $j$ and $i \ge 0$, $\Delta_{\beta_{j,i+1}-r_j} \le 2^k \Delta_{\beta_{j,i}-r_j}$. Also $\Delta_l$ is non-decreasing in $l$.*

LEMMA 3.6. $\mathrm{E}[\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y^*)] - \mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y) \le \frac{2^{3k+3}}{c} \mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y)$.

*Proof.* To bound the cost, fix a job $j$ and an integer $i \ge 2$. If time $\beta_{j,i}$ is sampled to be $j$'s completion time, then $y^*_{j,t}$ will be set to 1 for all $r'_j \le t \le \beta_{j,i}$. The incurred cost is upper bounded by $\sum_{r'_j < t \le \beta_{j,i}} \Delta_{t-r_j} \le \sum_{r'_j < t \le \beta_{j,i}} \Delta_{\beta_{j,i}-r_j} = (\beta_{j,i} - r'_j)\Delta_{\beta_{j,i}-r_j} = 2^i \Delta_{\beta_{j,i}-r_j}$, where the first inequality follows from Proposition 3.2. The probability of $\beta_{j,i}$ being sampled for $i \ge 2$ is at most $\frac{1}{c} y_{j,\beta_{j,i-2}}$. Thus the expected cost is at most $\frac{1}{c} y_{j,\beta_{j,i-2}} 2^i \Delta_{\beta_{j,i}-r_j}$ due to $\beta_{j,i}$ being sampled as $j$'s completion time. Thus, by summing

[4] This is well-defined since $y_{j,b(I)} \le c \le 1/2$. To see this, for the sake of contradiction, suppose $y_{j,b(I)} \ge c$. Consider any $i \ge 2$ such that $\beta_{j,i-2} \le b(I) \le \beta_{j,i}$. Since $y_{j,\beta_{j,i-2}} \ge y_{j,b(I)}$, $\beta_{j,i}$ is sampled with probability 1, which forces $y_{j,b(I)} = 1$ satisfying Constraint (3.4) for $I$.

over all $i \geq 2$ and $j$ and by using the linearity of expectation, we upper bound the expected total increase of the integral LP cost by $\frac{1}{c}\sum_j \sum_{i\geq 2} 2^i y_{j,\beta_{j,i-2}} \Delta_{\beta_{j,i}-r_j} \leq \frac{1}{c}\cdot 4^{k+1} \sum_j \sum_{i\geq 0} 2^i y_{j,\beta_{j,i}} \Delta_{\beta_{j,i}-r_j}$. The inequality is due to Proposition 3.2. This, together with the following lower bound, yields the lemma.

$$
\begin{aligned}
&(3.5)\ \mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y) \\
&= \sum_j \Big( \Delta_{\beta_{j,0}-r_j} y_{j,\beta_{j,0}} + \sum_{i\geq 1} \sum_{t=\beta_{j,i-1}+1}^{\beta_{j,i}} \Delta_{t-r_j} y_{j,t} \Big) \\
&\geq \sum_j \Big( \Delta_{\beta_{j,0}-r_j} y_{j,\beta_{j,0}} + \sum_{i\geq 1} 2^{i-1} \Delta_{\beta_{j,i-1}-r_j} y_{j,\beta_{j,i}} \Big) \\
&\geq \frac{1}{2^{k+1}} \sum_j \Big( \Delta_{\beta_0-r_j} y_{j,\beta_{j,0}} + \sum_{i\geq 1} 2^i \Delta_{\beta_{j,i}-r_j} y_{j,\beta_{j,i}} \Big) \\
&\geq \frac{1}{2^{k+1}} \sum_j \sum_{i\geq 0} 2^i \Delta_{\beta_{j,i}-r_j} y_{j,\beta_{j,i}},
\end{aligned}
$$

where the first inequality follows by Proposition 3.2 and Constraints (3.3).

To show that the fractional LP cost decreases significantly in each iteration, we bound the probability that a fixed job $j$ is critical for some intervals whose constraints are unsatisfied after step (i). That is, the probability that $j \in N_I$ for an interval $I$ whose constraint is not satisfied by the randomized rounding. If we show that the probability is significantly small, then we will be able to obtain a new solution whose objective is much smaller than that of the original LP solution. This is done separately for intervals ending between $(\beta_{j,i-1}, \beta_{j,i}]$. Let $G_{i,j}$ be the set of intervals $I$ ending during $(\beta_{j,i-1}, \beta_{j,i}]$ such that $j \in N_I$. To make this well-defined, we include $I$ in $G_{i,j}$ only when $y_{j',b(I)} < c$ for all $j' \in H_I$; if not, Constraint (3.4) for $I$ is satisfied after step (i).

LEMMA 3.7. *Fix any job $j$ and any integer $i \geq 2$. The probability that* any *interval in $I \in G_{i,j}$ is unsatisfied by step (i) is at most* $\exp(-\frac{2}{5c})$.

*Proof.* Fix an interval $I' = [a(I'), b(I')]$ in $G_{i,j}$ that has the *latest* starting time. Note that this implies that $I'$ is the interval with the latest starting time such that $j \in N_{I'}$ and $\beta_{j,i-1} < b(I') \leq \beta_{j,i}$. We know that by definition of $N_{I'}$ there is a set $E_{I'}$ of jobs that arrive during $I'$ and no later than $j$ such that $\sum_{j'\in E_{I'}} y_{j',b(I')} \geq \frac{2}{5}$. Further, by Lemma 3.4 we know that $y$ is priority-preserving which implies that any job $j' \in E_{I'}$ with $y_{j',b(I')} > 0$ has $p_{j'} \geq p_j$. This is because $y_{j,b(I')} < 1$ since otherwise $I'$ is satisfied by job $j$, but then since $r_{j'} \leq r_j$ it must be that $p_{j'} \geq p_j$ due to the fact that $y$ is priority-preserving. Also for the same reason note that for any $j' \in E_{I'}$ with $y_{j',b(I')} > 0$, we have $r'_{j'} \leq r'_j$ since $y_{j',r'_{j'}} = 1$ by definition of $r'_{j'}$ and due to the fact that $y_{j,r'_{j'}} = 1$. See Figure 1 for a visualization of this setting.

Using the above properties, we want to show that if *any* job $j' \in E_{I'}$ with $y_{j',b(I')} > 0$ is given a completion time later than $\beta_{j,i}$ by step (i) then *every* interval in $G_{i,j}$ is satisfied. If we can show this and additionally show that some job $j' \in E_{I'}$ is given a completion time later than $\beta_{j,i}$ with a good probability then the lemma will follow.

To show this, we first need to show that every job $j' \in E_{I'}$ with $y_{j',b(I')} > 0$ has $r_{j'} \in I = [a(I), b(I)]$ for every interval $I$ in $G_{i,j}$. Fix any $I \in G_{i,j}$. We know that $r_j \leq b(I)$ since $I \in G_{i,j}$, thus $j \in N_I$, and $r_{j'} \leq r_j$ by definition of $E_{I'}$. Hence we have $r_{j'} \leq b(I)$. Now we show $a(I) \leq r_{j'}$. By definition of $I'$, we know that $I'$ starts no earlier than $I$ by definition of $I'$, i.e. $a(I) \leq a(I')$. Further, since $j' \in E_{I'}$, by definition of $E_{I'}$, $j'$ must arrive during $I'$, so we have $a(I') \leq r_{j'}$. Thus we have shown that $r_{j'} \in I = [a(I), b(I)]$ as desired. Further, we noted above that $p_{j'} \geq p_j$. Hence job $j'$ can be used to satisfy Constraint (3.4) for $I$ since job $j$ is sufficiently large to satisfy the constraint, and $p_{j'} \geq p_j$. Therefore, if any job $j'$ in $E_{I'}$ is given a completion time later than $\beta_{j,i}$ by Step (i) then the constraint for $I$ is satisfied. This follows from knowing that every interval $I \in G_{i,j}$ ends no later than $\beta_{j,i}$, $r_{j'} \in I = [a(I), b(I)]$ and $j'$ is large enough to satisfy Constraint (3.4) for $I$.

It only remains to bound the probability that no job in $E_{I'}$ is given a deadline after $\beta_{j,i}$. Consider that any job $j'$ in $E_{I'}$ with $y_{j',b(I')} > 0$. Let $i'$ be an integer such that $\beta_{j',i'-1} < \beta_{j,i} \leq \beta_{j',i'}$. Since $r'_{j'} \leq r'_j$, it must be the case that $i' \geq i$, which implies $\beta_{j',i'-2} \leq \beta_{j,i-1}$ And we know that completion time $\beta_{j',i'}$ is sampled for job $j'$ with probability $\frac{1}{c} y_{j',\beta_{j',i'-2}}$, which is greater than $\frac{1}{c} y_{j',\beta_{j,i-1}} \geq \frac{1}{c} y_{j',b(I')}$ due to Constraint (3.3); note that $y_{j',b(I')} < c$ since $I' \in G_{j,i}$. Thus, the probability that no job in $E_{I'}$ has a completion time sampled no earlier than $\beta_{j,i}$ is at most $\prod_{j\in E_{I'}} (1 - y_{j',b(I')}/c) \leq \prod_{j'\in E_{I'}} \exp(-y_{j',b(I')}/c) \leq \exp(\sum_{j'\in E_{I'}} -y_{j',b(I')}/c) \leq \exp(-\frac{2}{5c})$.

The previous lemma will allow us to bound the fractional LP cost in the new LP solution $y^*$.

LEMMA 3.8. $\mathrm{E}[\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y^*)] \leq 2^{2k+3} \exp(-(\frac{2}{5c}))\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y)$.

*Proof.* To prove the lemma, we first bound the cost for each job $j$. Consider the case that an interval $I' = [s', t']$ in $G_{i,j}$ is unsatisfied after step (i). This could increases $y^*_{j,t}$ by at most $2y_{j,t'}$ for all $r'_j < t \leq t'$. Note that $y_{j,t'} \leq y_{j,\beta_{j,i-1}}$ by Constraint (3.3). The total cost incurred due to this operation is at most $2\sum_{r'_j < t \leq \beta_{j,i}} \Delta_{t-r_j} y_{j,\beta_{j,i-1}} \leq 2^{i+1} \Delta_{\beta_{j,i}-r_j} \cdot y_{j,\beta_{j,i-1}}$. By summing over all $i \geq 2$

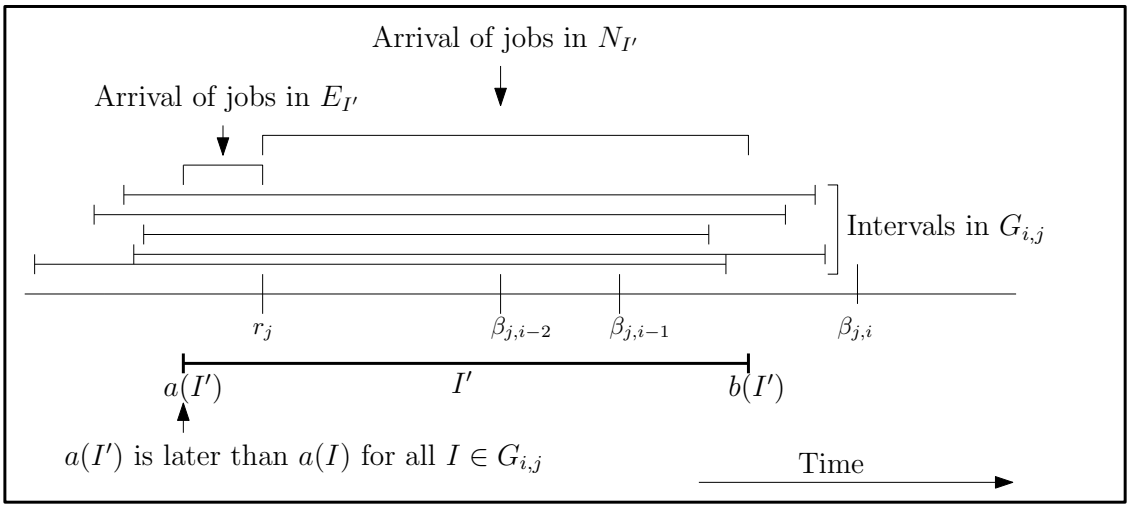


Figure 1: Intervals in $G_{i,j}$. Note that all jobs in $E_{I'}$ arrive during every interval in $G_{i,j}$. Due to this, if a job in $E_{I'}$ is given a completion time later than $\beta_{j,i}$ then all intervals in $G_{i,j}$ are satisfied.

and $j$ and using the linearity of expectation, the expected increase of the cost is at most

$$\begin{aligned}
&2\sum_j \sum_{i\geq 2} 2^i \Delta_{\beta_{j,i}-r_j} \cdot 2y_{j,\beta_{j,i-1}} \\
&\leq 4\sum_j \sum_{i\geq 1} 2^i \Delta_{\beta_{j,i+1}-r_j} \cdot y_{j,\beta_{j,i}} \\
&\leq 2^{k+2} \sum_j \sum_{i\geq 1} 2^i \Delta_{\beta_{j,i}-r_j} \cdot y_{j,\beta_{j,i}}
\end{aligned}$$

by Proposition 3.2. This upper bound, together with (3.5) and Lemma 3.7, yields the lemma.

As mentioned before, the solution $y^*$ we obtained by performing steps (i) and (ii) is not necessarily priority-preserving. If it is, then Lemmas 3.6 and 3.8 immediately imply Lemma 3.3. In Appendix A we show how to ensure that $y^*$ is priority preserving, completing the proof.

## References


[1] Foto N. Afrati, Evripidis Bampis, Chandra Chekuri, David R. Karger, Claire Kenyon, Sanjeev Khanna, Ioannis Milis, Maurice Queyranne, Martin Skutella, Clifford Stein, and Maxim Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 32–44, 1999.

[2] Susanne Albers. On randomized online scheduling. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 134–143, 2002.

[3] S. Anand, Naveen Garg, and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *ACM-SIAM Syposium on Discrete Algorithms*, pages 1228–1241, 2012.

[4] Nikhil Bansal and Kedar Dhamdhere. Minimizing weighted flow time. *ACM Transactions on Algorithms*, 3(4), 2007.

[5] Nikhil Bansal and Janardhan Kulkarni. Minimizing flow-time on unrelated machines. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 851–860, 2015.

[6] Nikhil Bansal and Kirk Pruhs. Server scheduling to balance priorities, fairness, and average quality of service. *SIAM J. Comput.*, 39(7):3311–3335, 2010.

[7] Nikhil Bansal and Kirk Pruhs. The geometry of scheduling. *SIAM J. Comput.*, 43(5):1684–1698, 2014.

[8] Carl Bussema and Eric Torng. Greedy multiprocessor server scheduling. *Oper. Res. Lett.*, 34(4):451–458, 2006.

[9] Robert D. Carr, Lisa Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, pages 106–115, 2000.

[10] Jivitej S. Chadha, Naveen Garg, Amit Kumar, and V. N. Muralidhara. A competitive algorithm for minimizing weighted flow time on unrelatedmachines with speed augmentation. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 679–684, 2009.

[11] Timothy M. Chan, Elyot Grant, Jochen Könemann, and Malcolm Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto,*

*Japan, January 17-19, 2012*, pages 1576–1585, 2012.

[12] Chandra Chekuri, Ashish Goel, Sanjeev Khanna, and Amit Kumar. Multi-processor scheduling to minimize flow time with epsilon resource augmentation. In *ACM Symposium on Theory of Computing*, pages 363–372, 2004.

[13] Chandra Chekuri, Sungjin Im, and Benjamin Moseley. Longest wait first for broadcast scheduling [extended abstract]. In *WAOA*, pages 62–74, 2009.

[14] Chandra Chekuri and Sanjeev Khanna. A PTAS for minimizing weighted completion time on uniformly related machines. In *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, pages 848–861, 2001.

[15] Chandra Chekuri and Sanjeev Khanna. Approximation schemes for preemptive weighted flow time. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 297–305, 2002.

[16] Chandra Chekuri, Rajeev Motwani, B. Natarajan, and Clifford Stein. Approximation techniques for average completion time scheduling. *SIAM J. Comput.*, 31(1):146–166, 2001.

[17] Jeff Edmonds, Sungjin Im, and Benjamin Moseley. Online scalable scheduling for the ;k-norms of flow time without conservation of work. In *ACM-SIAM Syposium on Discrete Algorithms*, pages 109–119, 2011.

[18] Jeff Edmonds and Kirk Pruhs. Scalably scheduling processes with arbitrary speedup curves. *ACM Transactions on Algorithms*, 8(3):28, 2012.

[19] Kyle Fox and Benjamin Moseley. Online scheduling on identical machines using srpt. In *ACM-SIAM Syposium on Discrete Algorithms*, pages 120–128, 2011.

[20] Naveen Garg and Amit Kumar. Minimizing average flow time on related machines. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 730–738, 2006.

[21] Anupam Gupta, Sungjin Im, Ravishankar Krishnaswamy, Benjamin Moseley, and Kirk Pruhs. Scheduling jobs with varying parallelizability to reduce variance. In *Symposium on Parallelism in Algorithms and Architectures*, pages 11–20, 2010.

[22] Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. In *Workshop on Approximation and Online Algorithms*, 2012.

[23] Dorit S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., Boston, MA, USA, 1997.

[24] Sungjin Im and Benjamin Moseley. Online scalable algorithm for minimizing $\ell_k$-norms of weighted flow time on unrelated machines. In *ACM-SIAM Syposium on Discrete Algorithms*, pages 95–108, 2011.

[25] J. Labetoulle, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Preemptive scheduling of uniform machines subject to release dates. *Progress in combinatorial optimization*, January 1982.

[26] Joseph Leung, Laurie Kelly, and James H. Anderson. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, Inc., Boca Raton, FL, USA, 2004.

[27] Benjamin Moseley, Kirk Pruhs, and Cliff Stein. The complexity of scheduling for p-norms of flow and stretch - (extended abstract). In *Integer Programming and Combinatorial Optimization - 16th International Conference, IPCO 2013, Valparaíso, Chile, March 18-20, 2013. Proceedings*, pages 278–289, 2013.

[28] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 3rd edition, 2008.

[29] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice Hall Press, Upper Saddle River, NJ, USA, 2007.

[30] Kasturi R. Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 641–648, 2010.

## A Ensuring $y^*$ is Priority-Preserving

Suppose $y^*$ is not priority-preserving. If there are two jobs $i$ and $j$ where $r_j \leq r_i$ and $p_j < p_i$ however there is a time $t \geq r_i$ where $y^*_{i,t} < 1$ and $y^*_{j,t} > 0$, we say this is a priority inversion. We first observe that priority inversions appearing in solution $y^*$ have special forms, which will help make us make $y^*$ priority-preserving. Towards this end, we set up some notations. We define a partial order amongst jobs: $j \prec i$ if $r_j \leq r_i$ and $p_j < p_i$. Note that the partial order is transitive and defines a DAG amongst jobs. We say that two jobs $j \prec i$ preserve priorities at time $\tau$ for solution a $y'$ if $y'_{i,\tau} < 1$ implies $y'_{j,\tau} = 0$, and denote it as $j \prec_{y',\tau} i$. If not, we denote it as $j \not\prec_{y',\tau} i$. Note that $y'$ being priority-preserving means that for all $j \prec i$, $j \prec_{y',\tau} i$ for all times $\tau > r_i$. The following lemma states that if at any time $\tau$, a lower-priority job $i$ does not have $y^*_{i,t} = 1$ while a higher-priority job $j$ has $y^*_{j,t} > 0$, then it must be the case that job $j$ has $y^*_{j,t} = 1$.

LEMMA A.1. *After steps (i) and (ii), suppose that there are two jobs $j \prec i$ such that $j \not\prec_{y^*,\tau} i$ for a time $\tau > r_i$. Then it must be the case that $y^*_{j,\tau} = 1$. Further, for all times $t \geq \tau$, $y^*_{j,t} \in \{0, 1\}$.*

*Proof.* We restrict our attention to a pair of jobs $j \prec i$ such that $j \not\prec_{y^*,\tau} i$ for some $\tau > r_i$. Since the solution $y$ given in Lemma 3.3 is priority-preserving, we have that $j \prec_{y,\tau} i$. We consider two cases. The first case is that a completion time no earlier than $\tau$ was sampled for job $j$. Then, after step (i), $y^*_{j,\tau} = 1$. It is easy to see that $y^*_{j,\tau}$ remains to have value 1 after step (ii) since step (ii) doesn't decrease $y^*$ values. We also observe that $y_{j,\tau} = 0$. Otherwise, since $j \prec_{y,\tau} i$, it follows that $y_{i,\tau} = 1$. Then it must be the case that $y^*_{i,\tau} = 1$, meaning that $j \prec_{y^*,\tau} i$, which is a contradiction. Hence for all $t \geq \tau$, we have $y_{j,t} = 0$. Thus the lemma follows in the case.

Now consider the other case which we will show

never occurs. Since $j \not\prec_{y^*,\tau} i$, after step (ii) we have $y^*_{i,\tau} < 1$. Since once a $y^*$ variable becomes 1 during steps (i) or (ii), it remains to be 1 until the end of these steps, it must be the case that $y_{i,\tau} < 1$. Since $j \prec_{y,\tau} i$, it must be the case that $y_{j,\tau} = 0$. This implies that $y^*_{j,\tau} = 1$ since $y^*_{j,\tau} > 0$ and step (ii) does not make a zero-valued $y$ variable to become non-zero, and step (i) can only make $y^*_{j,\tau}$ become 1. But this means that job $j$ was given a completion time no earlier than $\tau$ in step (i), which is a contradiction to the assumption of this case. Hence only the first case can occur for which we showed the lemma.

Note that the previous lemma implies any job $j$ only has integral variables if there is a job $i$ and $t$ such that $j \not\prec_{y^*,\tau} i$.

Using the previous lemma, we now define a procedure to convert $y^*$ so that it becomes priority-preserving. To do this, we define a new solution $\bar{y}$. Initially we set $\bar{y}_{j,t} = y_{j,t}$. We recursively do the following procedure until there are no more priority inversions in the solution $\bar{y}$. Consider the job $j$ that arrives the earliest such that there is a job $i$ and time $t$ where $j \not\prec_{\bar{y},t} i$; in case of ties, choose a job with the smallest size. Fix this job $j$ and let $T$ be the set of times $t$ where there exists a job $i$ and $j \not\prec_{\bar{y},t} i$. Consider times $t \in T$ in increasing order and consider a job $i$ with the maximum $r_i$ such that $j \not\prec_{\bar{y},t} i$; in case of ties, choose a job with the largest size. Set $\bar{y}_{i,t} = 1$ and $\bar{y}_{j,t} = 0$. This is performed for all times in $T$. The algorithm then recurses on another such job $j$ so long as there are priority inversions. We will show that this algorithm must only reduce the number of priority inversions, implying the procedure will terminate in polynomial time.

We first show that this procedure does not increase the objective value. We will then show that the solution remains a feasible solution. Finally, we show that the procedure does not introduce any new priority inversions and, therefore, will eventually terminate.

LEMMA A.2. *After performing a modification to $\bar{y}$ no new priority inversions are introduced.*

*Proof.* For the sake of contradiction consider the first time we introduce a priority inversion when we were considering jobs $j$ and $i$ and a time $t$ such that $j \not\prec_{\bar{y},t} i$. There are two reasons this could happen. The first is because we set $\bar{y}_{i,t}$ to be 1. For this to create an inversion, there must be a job $i' \succ i$ such that $\bar{y}_{i',t} < 1$. However, then before this operation we had $j \not\prec_{\bar{y},t} i'$ because $j \prec i \prec i'$. This implies that $r_{i'} > r_i$, or $p_{i'} > p_i$ and $r_{i'} = r_i$, which contradicts the way we chose $i$.

The second reason there could be an inversion is because we set $\bar{y}_{j,t} = 0$. In this case, it must be that there is a job $j' \prec j$ such that $\bar{y}_{j',t} > 0$. However, then $j' \prec j \prec i$. This implies that $j' \not\prec_{\bar{y},t} i$ contradicting the assumption that $j$ is a job with the smallest size amongst all jobs with the earliest arrival times that have a priority inversion at any time.

Now we bound the cost of the operations.

LEMMA A.3. *The modification of $y^*$ does not increase the objective of either $\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y^*)$ or $\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y^*)$.*

*Proof.* We know $\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y^*) = \mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(\bar{y})$ and $\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y^*) = \mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(\bar{y})$ before the modification. Note when considering job $j$, job $i$ and time $t$ we decrease $\bar{y}_{j,t}$ from 1 to 0 and increase $\bar{y}_{i,t}$ from at smallest 0 to 1. It must be the case that $\bar{y}_{j,t}$ was 1 because by Lemma A.1 if there is an inversion $j \not\prec_{\bar{y},t} i$ then $\bar{y}_{j,t}$ is 1 before we perform the modification. Further, we introduce no new inversions by Lemma A.2. Now consider the change in the objective in this operation. The change of the value is upper bounded by $-\Delta_{\tau-r_j} + \Delta_{\tau-r_i}$ for the integral cost, which is non-positive since $r_j \leq r_i$. For fractional variables, we can only make them integral so the fractional cost cannot increase.

Finally, we show that the new solution is feasible.

LEMMA A.4. *The modified solution $\bar{y}$ at the end of the above procedure is feasible.*

*Proof.* We consider constraints (3.3) and (3.4) separately. First consider (3.3). For the sake of contradiction, consider the jobs $j$ and $i$ and time $t$ which is the first time the algorithm makes the solution $\bar{y}$ infeasible for a constraint in (3.3). This can be caused by two cases. For the first case, say this is because we set $\bar{y}_{i,t} = 1$. Then we know that $t > r_i$ and there must be a $t'$ where $\bar{y}_{i,t'} < 1$ and $r_i \leq t' < t$. This implies that at time $t'$ the algorithm had $j \not\prec_{\bar{y},t'} i$. The algorithm considers times in increasing order, so when the algorithm considered time $t'$ there must have been another job $i'$ where $j \not\prec_{\bar{y},t'} i'$, and $i \prec i'$ (i.e. $r_i < r_{i'}$, or $r_i = r_{i'}$ and $p_i < p_{i'}$). Otherwise, the algorithm would have set $\bar{y}_{i,t'} = 1$. However, then $j \not\prec_{\bar{y},t} i'$ and $r_{i'} > r_i$ contradicting the algorithm choosing a largest one amongst jobs with the latest arrival times that $j$ has a priority inversion with at $t$.

For the second case say that the solution becomes infeasible due to setting $\bar{y}_{j,t} = 0$. This could indeed make the solution infeasible if there is a later time $t' > t$ where $\bar{y}_{j,t'} > 0$. However, we know that for any $t' \geq t$ that if $\bar{y}_{j,t'} > 0$ then $j$ is in an inversion with $i$ as the solution was feasible before we considered job $j$, thus $\bar{y}_{i,t'} < \bar{y}_{i,t} < 1$. Thus, after we are done with considering steps for $j$, it will be the case that for all $t' \geq t$ we have $\bar{y}_{j,t'} = 0$.

Finally we show that constraints (3.4) are satisfied. Consider the first time a constraint in (3.4) becomes unsatisfied for an interval $I$ when we considered the jobs $j$ and $i$ and time $t$. This constraint must become infeasible

due to setting $\bar{y}_{j,t} = 0$. Thus, we know that $j \in H_I$ so then $a(I) \le r_j \le t = b(I)$. We also know by definition of an inversion that $p_j < p_i$ and $r_j \le r_i \le t$. Thus $i$ has a bigger size than $j$, and $i$ also arrives during $I$ and therefore $i \in H_I$ by definition of $H_I$. However, then we set $\bar{y}_{i,t} = 1$, so the constraint must be satisfied.

The previous lemmas complete the proof of the claim in Lemma 3.3 that the new solution is priority-preserving.

## B Light Intervals

In this section, we show that we can set jobs completion times so that all constraints for light intervals are satisfied while keeping the objective to be at most $\frac{O(1)^k}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$. As mentioned before, this rounding closely follows [7], but for completeness we include the rounding procedure along with the analysis.

**B.1 Preprocessing the LP solution** We first preprocess the solution to $\mathsf{LP}_{\mathsf{main}}$ we had after the threshold rounding in Section 2 to make the analysis easier. Recall that after the threshold rounding, we partitioned intervals left unsatisfied into two groups, $\mathcal{H}$ and $\mathcal{L}$, and we wanted to satisfy Constraints (2.2) for all light intervals, $\mathcal{L}$. We had an assignment $\tilde{x}_{j,t}$ such that $\sum_{j\in L_I} p_j \tilde{x}_{j,b(I)} \ge \frac{1}{2}(V(I) - |I| - P(S_{c,I}))$ for all $I \in \mathcal{L}$ where for every $j \in L_I$, $p_j < V(I) - |I| - P(S_{c,I})$ and $\tilde{x}_{j,b(I)} < c$. We first increase all $\tilde{x}_{j,t}$ such that $\tilde{x}_{j,t} < c$ by a factor of 16. For each job $j$ and $i \ge 0$, define $\beta_{j,i} := r_j + 2^i$; also define $\beta_{j,-1} = r_j$. Then, for each $j$ and $i \ge 1$ for all times $t \in (\beta_{j,i-1}, \beta_{j,i}]$, we set $\tilde{x}_{j,t} = \tilde{x}_{j,\beta_{j,i-1}}$. This only increases the value of every variable $\tilde{x}_{j,t}$ since $\tilde{x}$ satisfies Constraints (2.2), thus keeping the solution feasible. It is straightforward to see that other constraints remain satisfied. Further, using the fact that $(\beta_{j,i+1} - r_j)^k \le 2^k(\beta_{j,i} - r_j)^k$, we can also show that the LP cost doesn't increase a lot.

LEMMA B.1. *The LP solution remains feasible and the cost increases by a factor of at most* $2^{3k+4}$.

*Proof.* Since we already verified that the new $\tilde{x}$ satisfies all constraints, we only show the second claim. The first step increases the objective by a factor of at most 16 as mentioned. Before the second step, the objective is at least

$$\begin{aligned}&\Delta_{\beta_{j,0}-r_j}\tilde{x}_{j,\beta_{j,0}} + \sum_{i\ge1}\sum_{t=\beta_{j,i-1}+1}^{\beta_{j,i}} \Delta_{t-r_j}\tilde{x}_{j,\beta_{j,i}}\\ &\ge \Delta_{\beta_{j,0}-r_j}\tilde{x}_{j,\beta_{j,0}} \\ &\quad + \sum_{i\ge1}(2^{ik} - 2^{(i-1)k})\Delta_{\beta_{j,i-1}-r_j}\tilde{x}_{j,\beta_{j,i}}.\end{aligned}$$

Similarly, one can see that the objective increases by at most

$$\begin{aligned}&\sum_{i\ge1}\sum_{t=\beta_{j,i-1}+1}^{\beta_{j,i}} \Delta_{t-r_j}\tilde{x}_{j,\beta_{j,i-1}}\\ &\le \sum_{i\ge1}(2^{ik} - 2^{(i-1)k})\Delta_{\beta_{j,i}-r_j}\tilde{x}_{j,\beta_{j,i-1}}\\ &\le 4^k\Delta_{\beta_{j,0}-r_j}\tilde{x}_{j,\beta_{j,0}} \\ &\quad + \sum_{i\ge2}(2^{ik} - 2^{(i-1)k})\Delta_{\beta_{j,i}-r_j}\tilde{x}_{j,\beta_{j,i-1}}\\ &\le 4^k\Delta_{\beta_{j,0}-r_j}\tilde{x}_{j,\beta_{j,0}} \\ &\quad + 2^k\sum_{i\ge1}(2^{ik} - 2^{(i-1)k})\Delta_{\beta_{j,i+1}-r_j}\tilde{x}_{j,\beta_{j,i}}\end{aligned}$$

By using the fact that $\Delta_{\beta_{j,i+1}-r_j} \le 4^k\Delta_{\beta_{j,i-1}-r_j}$ and factoring in the factor 16 increase we had due to the first step, we obtain the desired lemma.

Let $D_I = (V(I) - |I| - P(S_{c,I}))$ be the residual demand for interval $I$. Now we have a fractional solution $x$ such that for every light interval $I \in \mathcal{L}$, $\sum_{j\in L_I} p_j\tilde{x}_{j,b(I)} \ge 8D_I$. We want to find an integral solution $x$ that satisfies all constraints, particularly Constraint (2.2) for each light interval $I$ with residual demand $D_I$. From now on, for simplicity, we will assume that every job have a size equal to a power of two. This can be done by rounding up each job size to the nearest power of two. Then it suffices to find a solution that satisfies Constraint (2.2) for each interval $I$ in $\mathcal{L}$ with demands $2D_I$. Since for each job $j$ we partitioned times after $j$'s arrival into $(\beta_{j,i}, \beta_{j,i+1}]$, $i \ge 0$, and we will require $x_{j,t}$ has the same value for all $t \in (\beta_{j,i}, \beta_{j,i+1}]$, it will be convenient to further simplify the LP by introducing new variables, $z_{j,i}$. Let $i_0(j)$ be the smallest $i$ such that we set $\tilde{x}_{j,\beta_{j,i}} < 1$. Since we will fix $x_{j,\beta_{j,i}} = 1$ for all $i < i_0(j)$, we will only consider variables $z_{j,i}$, $i \ge i_0(j)$; $z_{j,i_0(j)-1}$ is kept for notational convenience.

To summarize the above, every job has a size equal to a power of two, and for the following LP,

$$\begin{aligned}
(\mathsf{LP}_{\mathsf{light}}) \qquad & \min \sum_j \sum_{i\ge i_0(j)} \Delta'_{j,i} z_{j,i} \\
(\text{B.1}) \quad s.t. \quad & \sum_{j:r_j\in I, p_j<D_I} p_j \cdot z_{j,i(j,I)} \ge 2D_I && \forall I \in \mathcal{L} \\
(\text{B.2}) \quad & z_{j,i} \le z_{j,i-1} && \forall j, i \ge i_0(j) \\
(\text{B.3}) \quad & z_{j,i_0(j)-1} = 1 && \forall j \\
& z_{j,i} \ge 0 && \forall j, i \ge i_0(j),
\end{aligned}$$

where $\Delta'_{j,i} = (\beta_{j,i} - r_j)^k - (\beta_{j,i-1} - r_j)^k$ and $i(j, I)$ is the smallest $i'$ such that $\beta_{j,i'} \ge b(I)$, we have a feasible solution $\tilde{z}_{i,j}$ whose cost is at most $\frac{O(1)^k}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$

(see Claim 2.2 and Lemma B.1) such that

$$\sum_{j:r_j\in I, p_j<D_I} \tilde{z}_{j,i(j,I)} \geq 8D_I \tag{B.4}$$

Notice that in the summation of Constraint (B.1) we don't have the condition $j \notin S_{c,I}$. It is true that $j \in L_I$ if and only if $j$ arrives during $I$, $p_j < D_j$, and $j \notin S_{c,I}$. However, $j \in S_{c,I}$ only when $\tilde{x}_{j,\beta_{j,i(j,I)}} = 1$, and we don't even consider such pair $(j, i(j,I))$ since $i(j,I) < i_0(j)$ by definition of $i_0(j)$.

In the following section, we will show that one can find an integral solution $\bar{z}_{i,j}$ to $\mathsf{LP}_{\mathsf{light}}$ whose cost is $O(1)$ times the cost of $\tilde{z}_{i,j}$ for $\mathsf{LP}_{\mathsf{light}}$, $\mathsf{LP}_{\mathsf{light}}(\tilde{z})$. We will set $x_{j,t} = 1$ for all $r_j \leq t \leq \beta_{j,i}$ if $\bar{z}_{j,i} = 1$. Fix a job $j$ and let $i'$ be the largest $i$ such that $\bar{z}_{j,i} = 1$. This implies that $\bar{z}_{j,i_0(j)-1} = \bar{z}_{j,i_0(j)} = ... = \bar{z}_{j,i'} = 1$. So job $j$'s $k$th power of flow time will be $(\beta_{j,i'} - r_j)^k$. Part of it, till time $i_0(j)$, was already taken care of in the threshold rounding in Section 2 and Lemma B.1. The remaining part, $(\beta_{j,i'} - r_j)^k - (\beta_{j,i_0(j)} - r_j)^k$, is exactly the cost of the $\mathsf{LP}_{\mathsf{light}}$ due to job $j$ under the solution $\bar{z}$. Hence, an integral solution $\bar{z}_{i,j}$ to $\mathsf{LP}_{\mathsf{light}}$ will imply an integral solution to $\mathsf{LP}_{\mathsf{main}}$ with cost of $\frac{O(1)^k}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$ that satisfies all constraints for light intervals.

**B.2 Reduction to a Geometric Covering Problem** We now make a connection between $\mathsf{LP}_{\mathsf{light}}$ and the following geometric covering problem. We use $[x'_1, x'_2] \times [y'_1, y'_2]$ to denote an axis-parallel rectangle with points $(x'_1, y'_1), (x'_1, y'_2), (x'_2, y'_1)$ and $(x'_2, y'_2)$.

**The R2M problem.** The input is a collection $\mathcal{Q}$ of points in 2D Euclidean space, each with an integer demand $D_q$, and a collection of axis-parallel rectangles $B$ of the form $[0, x_B] \times [y^1_B, y^2_B]$, each with cost $w_B$. The goal is to find a minimum cost subset $S$ of rectangles such that each point $q \in \mathcal{Q}$ is covered by at least $D_q$ rectangles in $S$.

We will show that finding an integral solution to $\mathsf{LP}_{\mathsf{light}}$ can be reduced to the R2M problem. We create several instances of the R2M problem, $\mathcal{I}_\ell$, $0 \leq \ell \leq \log_2 \max_j p_j$. Each instance $\mathcal{I}_\ell$ has rectangles corresponding to jobs of size $2^\ell$. We create $\mathcal{I}_\ell$ as follows. For each interval $I = [a(I), b(I)] \in \mathcal{L}$, create a point $q_I = (a(I), b(I))$. Its demand $D_{I,\ell}$ will follow soon. For each job $j$ of size $2^\ell$, create a collection of rectangles $\{B_{j,i}\}_{i \geq i_0(j)}$ where $B_{j,i}$ denotes rectangle, $[0, r_j] \times (\beta_{j,i-1}, \beta_{j,i}]$. We set $D_{I,\ell} = \lfloor \sum_{j,i:p_j=2^\ell, q_I \in B_{j,i}} \tilde{z}_{j,i} \rfloor$. This completes the description of R2M instances, $\{\mathcal{I}_l\}$.

It was shown that the R2M problem admits a constant approximation [7]. Further, the approximation is based on a standard LP. The following IP exactly captures the R2M problem for instance $\mathcal{I}_\ell$ where variable $z'_{j,i} = 1$ if and only if $B_{j,i}$ is chosen.

$$(\mathsf{IP}_\ell) \qquad \min \sum_{j:p_j=2^\ell} \sum_{i \geq i_0(j)} \Delta'_{j,i} z'_{j,i}$$

$$s.t \sum_{j,i:q_I \in B_{j,i}, p_j=2^\ell} z'_{j,i} \geq D_{I,\ell} \qquad \forall I \in \mathcal{L} \tag{B.5}$$

$$z'_{j,i} \in \{0,1\}$$

We get a LP relaxation $\mathsf{LP}_\ell$ by allowing $z'_{j,i} \in [0,1]$. We observe that $\tilde{z}_{j,i}$ is a feasible LP solution for the following reason. Observe that $q_I \in B_{j,i}$ if and only if $a(I) \leq r_j$ and $\beta_{j,i-1} < b(I) \leq \beta_{j,i}$. By definition of $D_{I,\ell}$, we have $D_{I,\ell} \leq \sum_{j,i:p_j=2^\ell, q_I \in B_{j,i}} \tilde{z}_{j,i}$, which implies that $\tilde{z}_{j,i}$ satisfies Constraints (B.5).

Let $\{\bar{z}'_{j,i}\}_{j:p_j=2^\ell}$ be an integral solution to $\mathsf{LP}_\ell$ we obtain using a constant approximation based on LP relaxation. Hence we have $\mathsf{LP}_\ell(\bar{z}') = O(1)\mathsf{LP}_\ell(\tilde{z})$. Initially we set $\bar{z}_{j,i} = 0$, and if $\bar{z}'_{j,i} = 1$, then set $\bar{z}_{j,i_0(j)} = \bar{z}_{j,i_0(j)+1} = ... = \bar{z}_{j,i} = 1$.

LEMMA B.2. $\mathsf{LP}_{\mathsf{light}}(\tilde{z}) = \frac{O(1)^k}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$.

*Proof.* Knowing that $\Delta'_{j,i} \leq \frac{1}{2}\Delta'_{j,i+1}$, we have $\mathsf{LP}_\ell(\bar{z}) = O(1)\mathsf{LP}_\ell(\bar{z}')$. Also, since the variables of $\mathsf{LP}_\ell$ are only defined over jobs of size $2^\ell$, we have $\mathsf{LP}_{\mathsf{light}}(\bar{z}) = \sum_\ell \mathsf{LP}_\ell(\bar{z}) = O(1)\sum_\ell \mathsf{LP}_\ell(\bar{z}') = O(1)\sum_\ell \mathsf{LP}_\ell(\tilde{z}) = O(1)\mathsf{LP}_{\mathsf{light}}(\tilde{z}) = \frac{O(1)^k}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$.

Now we show that $\bar{z}$ is feasible to $\mathsf{LP}_{\mathsf{light}}$. It is easy to see that $\bar{z}$ satisfies Constraints (B.2) and (B.3). It only remains to show that $\bar{z}$ satisfies Constraint (B.1) for each interval $I \in \mathcal{L}$.

$$\begin{aligned}
&\sum_{j:r_j\in I, p_j<D_I} p_j \cdot \bar{z}_{j,i(j,I)} \\
&\geq \sum_{\ell:2^\ell<D_I} 2^\ell \sum_{j:r_j\in I, p_j=2^\ell} \bar{z}'_{j,i(j,I)} \\
&= \sum_{\ell:2^\ell<D_I} 2^\ell \sum_{i,j:q_I\in B_{j,i}, p_j=2^\ell} \bar{z}'_{j,i} \\
&\geq \sum_{\ell:2^\ell<D_I} 2^\ell D_{I,\ell} \qquad [\bar{z}' \text{ is feasible to } \mathsf{LP}_\ell] \\
&= \sum_{\ell:2^\ell<D_I} 2^\ell \lfloor \sum_{j,i:q_I\in B_{j,i}, p_j=2^\ell} \tilde{z}_{j,i} \rfloor \\
&\geq \sum_{\ell:2^\ell<D_I} 2^\ell \sum_{j,i:q_I\in B_{i,j}, p_j=2^\ell} \tilde{z}_{j,i} - \sum_{\ell:2^\ell<D_I} 2^\ell \\
&\geq \sum_{j:r_j\in I, p_j<D_I} p_j \tilde{z}_{j,i(j,I)} - 2D_I \geq 6D_I \qquad \text{[Due to (B.4)]},
\end{aligned}$$

as desired. We used the fact that $r_j \in I$ and $\beta_{j,i-1} < b(I) = \beta_{j,i}$ if and only if $q_I \in B_{j,i}$, and the definition that $i(j,I)$ is $i'$ such that $\beta_{j,i'-1} < b(I) \leq \beta_{j,i'}$

## C When jobs sizes or release times are not polynomially bounded by $n$

In this section we show that we can assume w.l.o.g. that all job sizes and arrival times are polynomially bounded by $n$. Let $p_{\max} = \max_j p_j$. Let be $\epsilon > 0$ be an arbitrary small constant such that $1/\epsilon$ is an integer. We first observe that we can assume w.l.o.g. that no two jobs have arrival times that differ by more than $2np_{\max}$. Otherwise we can decompose the input instance into sub-instances where each sub-instance satisfies the simplifying assumption, and solve each separately. Further, our schedule will complete all jobs in each substance within $2np_{\max}$ times steps after the last job in the sub-instance arrives.

Hence we can assume w.l.o.g. that $\min_j r_j = 0$ and $\max_j r_j$ is polynomially bounded by $p_{\max}$ and $n$. Also assume that $p_{\max} \geq n^2/\epsilon^2$ since otherwise jobs sizes and arrival times are polynomially bounded by $n$. Define $\delta := \lfloor \frac{\epsilon^2 p_{\max}}{n^2} \rfloor$. Let $\mathcal{I}$ be the given instance. We modify $\mathcal{I}$ so that jobs arrival times and sizes are integer multiples of $\delta$. More precisely, for each job $j$, let $\bar{p}_j$ be an integer multiple of $\delta$ such that $p_j - 3\delta \leq \bar{p}_j \leq p_j - 2\delta$; if $p_j \leq 3\delta$, then $\bar{p}_j = 0$. Also let $\bar{r}_j$ be an integer multiple of $\delta$ such that $r_j \leq \bar{r}_j \leq r_j + \delta$. Let $\bar{\mathcal{I}}$ denote this new instance.

We first show that for any schedule $\sigma$ for $\mathcal{I}$, there is a schedule $\bar{\sigma}$ for $\bar{\mathcal{I}}$ where each job completes earlier than in schedule $\sigma$. To see this, notice that one can process job $j$ during $[r_j, \bar{r}_j]$ by at most $\delta$ units. However, job $j$'s size in $\bar{\mathcal{I}}$ is smaller than in $\mathcal{I}$ by at least $2\delta$. Hence even if we ignore the work that was done for job $j$ during $[r_j, \bar{r}_j]$ in schedule $\sigma$, we can complete job $j$ earlier if we assume that job $j$ has size $\bar{p}_j + \delta$.

Then, we scale down all parameters of $\mathcal{I}$, jobs arrival time and sizes uniformly by a factor of $\delta$, and solve it. Note that that all parameters in the resulting instance are polynomially bounded by $n$. To justify the scaling, it suffices to show that there is a nearly optimal schedule for $\bar{\mathcal{I}}$ where during any $[i\delta, (i+1)\delta]$ for integers $i$, no more than one job is processed. This can be easily shown by interpreting a schedule as a flow between intervals $\mathcal{R} = \{[i\delta, (i+1)\delta] : i \text{ is an integer}\}$ and jobs. Let $\bar{C}^*_j$ be $j$'s completion time of an optimal schedule for jobs $j$ with size $\bar{p}_j + \delta$ and arrival time $\bar{r}_j$. Then, we can view job $j$'s schedule as a flow of value $1 + \bar{p}_j/\delta$ from the job to intervals in $\mathcal{R}$ intersecting $[\bar{r}_j, \bar{C}^*_j]$. Using the integrality of flow, one can easily find a schedule where no more than one job is processed during any interval in $\mathcal{R}$, and each job $j$ completes within $\bar{C}^*_j + \delta$. Since this is a schedule for jobs with sizes $\bar{p}_j + \delta$, we can complete each job $j$ with size $\bar{p}_j$ by time $\bar{C}^*_j$.

Finally, we need to show that if we have a schedule for the modified instance $\bar{\mathcal{I}}$, then we can find a schedule for the original instance $\mathcal{I}$ without increasing the objective too much. Say $\bar{C}_j$ is $j$'s completion time for $\bar{\mathcal{I}}$. Then, it is easy to see that we can complete each job $j$ in $\mathcal{I}$ by time $\bar{C}_j + 3n\delta$: For notational simplicity, assume that jobs are ordered in increasing order of $\bar{C}_j$. Then, we can complete job 1 by time $\bar{C}_1 + p_1 - \bar{p}_1$, job 2 by time $\bar{C}_2 + (p_1 - \bar{p}_1) + (p_2 - \bar{p}_2)$, and so on. Let $F_j$ denote $j$'s flow time in $\mathcal{I}$, and let $\bar{F}_j$ denote the analogous quantity for instance $\bar{\mathcal{I}}$. Note that we have $F_j \leq \bar{F}_j + 3n\delta$.

LEMMA C.1. $F_j \leq \min\{(1+3)\bar{F}_j, 4\epsilon\frac{p_{\max}}{n}\}$.

*Proof.* To compare $\sum_j F_j$ and $\sum_j \bar{F}_j$, we consider two cases.
Case i): If $\bar{F}_j \geq \frac{\epsilon p_{\max}}{n}$. $F_j \leq \bar{F}_j + 3n\delta \leq F_j + 3\epsilon^2\frac{p_{\max}}{n} \leq (1+3\epsilon)\bar{F}_j$.
Case ii): Otherwise. $F_j \leq \bar{F}_j + 3n\delta \leq \epsilon\frac{p_{\max}}{n} + 3\epsilon^2\frac{p_{\max}}{n} \leq 4\epsilon\frac{p_{\max}}{n}$.
Hence the lemma follows.

By summing over all jobs, we have

$$\sum_j F_j^k \leq (1+3\epsilon)^k \sum_j \bar{F}_j^k + \frac{(4\epsilon)^k}{n^k} p_{\max}^k \cdot n$$
$$\leq (1+7\epsilon)^k \sum_j \bar{F}_j^k$$

The last inequality follows from the fact that the $k$th power of flow time is at least $p_{\max}^k$ for any schedule. Thus, we have shown that by solving the simplified instance $\bar{\mathcal{I}}$ where all parameters are polynomially bounded, we can construct a schedule for the original instance $\mathcal{I}$ whose cost is only $1 + 7\epsilon$ factor larger.

## D Omitted Proofs

**Proof of** [Lemma 2.1] To show the lemma, we simply need to show an algorithm that completes each job only earlier than $\mathsf{LP}_{\mathsf{main}}$ in the given solution $x$. We use the Earliest-Deadline-First algorithm (EDF), which always schedules an unsatisfied job with the earliest deadline at each time preemptively. Here the deadlines are the completion times given by the integral LP solution. Let $d_j$ be the deadline of job $j$. Note that $x_{j,r_j} = x_{j,r_j+1} = \ldots = x_{j,d_j} = 1$ and $x_{j,t} = 0$ for all $t > d_j$.

For the sake of contradiction, say that some job $j$ does not get scheduled by its deadline $d_j$. Let $t_1$ be the *earliest* time before $d_j$ such that at every time during the interval $[t_1, d_j]$ the EDF algorithm is always scheduling a job with deadline earlier than $d_j$; we can assume w.l.o.g. that jobs have distinct deadlines by breaking ties in an arbitrary but fixed way. Note that every job the algorithm schedules during $[t_1, d_j]$ arrives during $[t_1, d_j]$ since otherwise the value of $t_1$ must be smaller contradicting the definition of $t_1$. Consider Constraint (2.2) with $I = [t_1, d_j]$ and $S$ being the subset of jobs that arrive during $[t_1, d_j]$ but complete no earlier than $d_j$. Note that $V(I) - P(S)$

is the total size of jobs that arrive during $[t_1, d_j]$ and complete earlier than time $d_j$. Since EDF was always busy processing those jobs during $[t_1, d_j]$ but couldn't finish them all, it must be the case that $V(I) - P(S) > |I|$, which makes the right-hand-side of (2.2) strictly positive. On the other hand, for any job $j \in R(I) \setminus S$, we have $x_{j,d_j} = 0$ since the job completes earlier than time $d_j$. Hence the left-hand-side of (2.2) is 0, which is a contradiction. Thus all jobs are completed by their respective deadlines by EDF. □

**Proof of** [Claim 2.2] Fix a job $j$. Note that we changed $x_{j,t}$ only for times $t \in [r_j, C_{j,C}]$. The contribution of job $j$ to the LP objective during $[r_j, C_{j,C}]$ is at least $\frac{1}{c}(C_{j,c} - r_j)^k$. By setting $j$'s completion time to $C_{j,c}$, $j$'s contribution to the objective during $[r_j, C_{j,C}]$ increases to $(C_{j,c} - r_j)^k$, proving the claim. □