FAIR SCHEDULING VIA ITERATIVE QUASI-UNIFORM SAMPLING*

SUNGJIN IM† AND BENJAMIN MOSELEY‡

Abstract. This paper considers minimizing the ℓ_k-norms of flow time on a single machine offline using a preemptive scheduler for $k \geq 1$. The objective is ideal for optimizing jobs' overall waiting times while simultaneously being fair to individual jobs. This work gives 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. The main technical ingredient used in this work 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 [30, 33]. Typically there is a server, or alternatively a machine, which receives n requests for jobs to be processed on the machine. Each job j requires p_j units of 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. This work considers scheduling jobs that arrive over time on a single machine that are to be scheduled preemptively offline where every job j's arrival time r_j and processing time p_j are known in advance.

Given a scheduler, a job j is completed at the first time C_j when the scheduler has performed 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 [18, 16, 1, 2, 26].

While completion time objectives have been widely considered, when jobs arrive over time, more common metrics are based on the flow time of the jobs. The flow time of j is $C_j - r_j$, which is the total time job j waits to be completed after 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 a job 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 based on flow times is minimizing the total (or equivalently average) flow time $\sum_j (C_j - r_j)$. This objective focuses on optimizing the

*A preliminary version of this paper appeared in SODA 2017 [28].

† 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, CCF-1617653 and 1844939.

‡Tepper School of Business, Carnegie Mellon University, 5000 Forbes Av, Pittsburgh, PA 15213. moseleyb@andrew.cmu.edu. Supported in part by a Google Research Award, a Yahoo Research Award, a Bosch Junior Faculty Chair and NSF Grant CCF-1824303, 1830711, 1845146, 1733873 and CMMI-1938909.

average quality of service of the jobs or, alternatively, the average waiting time of jobs in 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 [34]. Due to the potential for unfairness, it is not surprising that some 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 ℓ_k-norm of the flow times $\sqrt[k]{\sum_j (C_j - r_j)^k}$. For the ℓ_k-norms, typically $k \in [2, 3]$, in which case the schedule is seeking to reduce 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 ℓ_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 [14, 11, 9, 22, 27], in broadcast scheduling [19, 15, 24], for parallel processors [19, 24] and on speed scalable processors [25].

The setting this work considers, preemptive job scheduling on a single machine, has been studied for decades. Despite being well researched, the complexity of some of 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 time and the ℓ_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 ℓ_k-norms of flow time admit an $O(1)$-approximation. It is well-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 [29] over three decades ago. The ℓ_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 $k \leq 2 < \infty$ [32]. 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. A recent breakthrough by Batra, Garg and Kumar [8] gave a $O(1)$-approximation for weighted flow time in pseudo-polynomial time and Feige, Kulkarni and Li [21] improve this to give a $O(1)$-approximation in polynomial time. If all jobs arrive at the same time, the shortest job first algorithm is optimal for the ℓ_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 [17]. Due to the existence of the pseudo-polynomial time and the 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 ℓ_k-norms of flow time is similar to weighted flow time, so it is additionally reasonable to believe this problem admits

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

an $O(1)$-approximation.

Similar techniques have been used to develop algorithms for both problems [20, 5, 4, 3, 23]. This is natural as the objectives are closely related. 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 ℓ_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 ℓ_2-norm objective of A, ignoring the outer square root, is $\int_{t=1}^{\infty} 2L^A(t)$. 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 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 ℓ_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 times. 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: This paper considers the ℓ_k-norms of flow time objective on a single machine. We show the following theorem.

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

To show the previous theorem, the analysis strongly uses a key property of the ℓ_k-norms of flow time, which does not hold in the case of average weighted flow time even for the ℓ_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 ℓ_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 that allows us to apply our algorithm and analysis framework.

We note that this property, however, does not hold in the weighted case even for the ℓ_1-norm. In particular, even 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 ℓ_k-norms of flow time an easier problem than weighted flow time.

1.1. An Overview of the Analysis Techniques. This section discusses the core techniques used at a high level. The most relevant work to ours is that of [7]. The work of [7] reduces the ℓ_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, this paper's approach is to consider the ℓ_k-norm problem directly.

We begin by using an integer program for the ℓ_k-norm problem, which was introduced in [7].[2] The integer program has a variable $x_{i,t}$ that is 1 if job i remains unsatisfied at time t and 0 otherwise. Thus, 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 there is a feasible schedule that finishes 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. The algorithm introduced in this work 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, the algorithm sets integral completion times of jobs to ensure all the constraints of the LP are satisfied. To do this, the 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, the algorithm ensures the completion times sampled only cost $O(1)$ more than the optimal LP cost. This procedure 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, the most natural strategy is to recurse and use iterative rounding to ensure the remaining constraints are satisfied. However, an algorithm 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 so the objective is bounded in the analysis.

To circumvent this hurdle the 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 ℓ_k-norm problem mentioned above. Intuitively, jobs in N_I are the most important jobs the LP used to satisfy the constraint for I. The algorithm recurses and utilizes iterative rounding; however, the algorithm only considers jobs that are in N_I for some interval I whose constraint is not satisfied. The analysis establishes that the probability that a fixed job is included in *any* set N_I over all intervals I that are unsatisfied 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.[3] 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, the approach is called *iterative quasi-uniform sampling*. After recursing $O(\log n)$ times, the analysis establishes that all intervals are satisfied, even though the recursion only considered

[2] In their paper, they conjecture that the linear program relaxation induced from the integer program have an $O(1)$-integrality gap for the general objective they consider, which encompasses total weighted flow time and ℓ_k-norms of flow time both.

[3] 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.

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

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

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

$$(2) \qquad \forall I = [s,t], \forall S \subseteq R(I) \text{ where } V(I) - |I| - P(S) \ge 0$$

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

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

FIG. 1. *A linear program relaxation for the kth power flow time objective.*

jobs in N_I to satisfy the constraint for I.

The rounding technique is in similar spirit as the quasi-uniform sampling technique of [35, 13] for geometric set cover; however, quasi-uniform sampling is not used for LP rounding and is a different algorithmic technique.

2. Preliminaries.

2.1. Formal Problem Definition and Notation. There is a set of n jobs that arrive over time. Each job j has an integer processing time/size p_j and arrives at some time $r_j \ge 0$. Both quantities $p_j \ge 1$ and r_j are assumed to be integers. Times are slotted. During each time slot $[t, t+1]$ for an integer $t \ge 0$, the machine can either process *one* job exactly by one unit or process no jobs. The goal is to schedule all jobs preemptively offline to minimize the ℓ_k norm of flow time. A *preemptive* scheduler can process a job a disjoint times. A job j completes at the earliest time when it is processed by p_j units after the jobs arrival.

If job j is completed at time C_j, the objective is $(\sum_j (C_j - r_j)^k)^{1/k}$. Define each job j's flow time to be $C_j - r_j$. Note that each job has flow time at least 1. Throughout the paper, the analysis will focus on minimizing the kth power flow time objective $\sum_j (C_j - r_j)^k$. This is equivalent to the ℓ_k-norms of flow time without the outer kth root. If a schedule is c^k-approximate for the kth power flow time objective, then it is a c-approximation for the kth norm of flow time objective for any $c \ge 1$. For an interval $I = [s,t]$ with integers $0 \le s \le t$, $|I|$ is defined as $t - s$, i.e., the number of time slots in the interval. 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 $I = [s,t]$.

2.2. Linear Programming Relaxation. A linear programming relaxation, $\mathsf{LP}_{\mathsf{main}}$, is now introduced for the kth power of flow time objective. See Figure 1 which was first established 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 that the number of time steps considered in the linear programming is polynomially bounded by n. This simplifying assumption will be removed in Section 5.

For completeness, the validity of $\mathsf{LP}_{\mathsf{main}}$ is discussed. To see why the LP is valid, 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 kth 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 time interval $I = [t_1, t_2]$. 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) 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, during 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) are standard knapsack covering inequalities [10] 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 obtain a solution arbitrarily close to the optimum; only the objective achieved is approximate and all constraints are satisfied exactly. It is shown in [7] that the LP can be solved in polynomial time as stated in the following theorem

THEOREM 2 ([7]). *There is a polynomial time algorithm that constructs a solution to the linear program (1) that is* $(1+\epsilon)$ *approximate in the objective and satisfies every constraint exactly.*

2.3. Linear Programing Rounding. The algorithm introduced in this work solves $\mathsf{LP}_{\mathsf{main}}$ and rounds the fractional solution to an integral solution. Once we have a feasible integral solution, it is easy to transform this into a feasible schedule of cost no more than the objective of the LP for this solution. The idea is to schedule each job no later than its completion time in the LP solution, by treating each job's completion time as its deadline. It is known that the Earliest-Deadline-First (EDF) algorithm will schedule all jobs by their deadline if there exists such a schedule. The proof is not difficult and follows by definition of EDF and constraints (2). The proof follows from [7] and is in the appendix for completeness.

LEMMA 3 ([7]). *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 fractional 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 the fractional solution to $\mathsf{LP}_{\mathsf{main}}$ and is feasible 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. Once the completion times are discovered, the corresponding LP solution sets $x_{j,t} = 1$ for all $t \in [r_j, C_j^*]$ and 0 otherwise.

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 on job j's completion time we set in all steps. Since completion times are only pushed back in each step, this ensures that a constraint in (2) satisfied by completion times in one step of the algorithm will remain satisfied at the end of the algorithm.

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 $r_j \leq t' \leq C_{j,c}$ owing to Constraints (1) — this remains the case regardless of the remaining rounding steps. After the threshold rounding, we will have a feasible solution to the LP still due to the knapsack cover inequalities. We note that the increase of the objective due to this rounding can easily be bounded by the cost of the LP. Let OPT(LP) denote the optimum for the LP. The proof of the following proposition is straightforward.

PROPOSITION 4. *After the threshold rounding, the LP cost is at most* $\frac{1}{c} \cdot \text{OPT}(\mathsf{LP}_{\mathsf{main}})$.

Proof of [Proposition 4] 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 fractional LP objective during $[r_j, C_{j,c}]$ is at least $c(C_{j,c} - r_j)^k$. That is $\sum_{t=r_j+1}^{C_{j,c}} x_{j,t}((t-r_j)^k - (t-1-r_j)^k) \geq c(C_{j,c}-r_j)^k$ because $x_{j,t} > c$ for $t \in [r_j, C_{j,c}]$. By setting j's completion time to $C_{j,c}$, j's contribution to the objective for times during $[r_j, C_{j,c}]$ increases to $(C_{j,c} - r_j)^k$, proving the proposition. □

For each interval I, let $S_{c,I}$ be the set of jobs that arrive during I that are given a deadline no earlier than the end of the interval I by the threshold rounding, i.e. $S_{c,I} := \{j \mid r_j \in I \text{ and } C_{j,c} \geq b(I)\}$. The threshold rounding may satisfy all constraints (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 $R(I) \setminus S_{c,I}$. Call an interval $I = [t_1, t_2]$ that is *not satisfied* by the threshold rounding **heavy** if $\sum_{j \in H_I} x_{j,t_2} \geq \frac{1}{2}$ and call the interval **light** otherwise; note that $\sum_{j \in L_I} p_j x_{j,t_2} \geq \frac{1}{2}(V(I) - |I| - P(S_{c,I}))$ if I is a light interval. Let $\mathcal{H}$ be the set of heavy intervals and $\mathcal{L}$ be the set of light intervals. We note that the idea of partitioning intervals is inspired by a similar partition done in [7] as well as for the related problem on column restricted covering [12].

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 both cases for each 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 is given in Section 4.

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 kth power of flow time. By taking the outer kth root for the ℓ_k norm objective, we obtain an $O(1)$ approximation for the ℓ_k norm by fixing c to be $\frac{1}{2^{k+4}}$.

3. Heavy Intervals. In this section, our goal is to set the completion times of jobs so that Constraints (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, $\mathsf{LP}_{\mathsf{main}}$. This is necessary as our rounding will crucially use the priority-preserving property (see Definition 6), which can be assumed w.l.o.g. when considering the relaxed LP. Hence the new LP, $\mathsf{LP}_{\mathsf{heavy}}$ will have an optimal solution cheaper than $\mathsf{LP}_{\mathsf{main}}$. The LP will be strong enough to give a solution that satisfies all constraints corresponding

to heavy intervals. Further, using a more relaxed LP will help iteratively construct feasible solutions to the LP. 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 $\Delta_{t-r_j} := (t-r_j)^k - (t-r_j-1)^k$.

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

$$\text{s.t.} \quad y_{j,t} \le y_{j,t-1} \qquad \forall j, t > r_j \tag{3}$$

$$\sum_{j \in H_I} y_{j,t} \ge 1 \qquad \forall I = [s,t] \in \mathcal{H} \tag{4}$$

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

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

$\mathsf{LP}_{\mathsf{heavy}}$ is a relaxation of $\mathsf{LP}_{\mathsf{main}}$. There are several changes. First we simplify Constraints (2) to only consider the heavy intervals; as mentioned, the constraints regarding light intervals are addressed in Section 4. Further, to satisfy the constraint for a heavy interval I, we will only use jobs in H_I. Next we discuss Constraints (4). 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 would have $\sum_{j \in H_I} y_{j,t} \ge 1$ for a heavy interval $I = [s,t] \in \mathcal{H}$ if and only if the completion time of at least one job in H_I is set to be at least t.

Proposition 5. $\mathrm{OPT}(\mathsf{LP}_{\mathsf{heavy}}) \le \frac{2}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$ *where* $\mathrm{OPT}(\mathsf{LP}_{\mathsf{heavy}})$ *is the optimal solution value of* $\mathsf{LP}_{\mathsf{heavy}}$ *and* $\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$ *is the optimal solution value of* $\mathsf{LP}_{\mathsf{main}}$.

Proof. Consider the fractional solution $\{x_{j,t}\}$ obtained in Section 2 after applying the threshold rounding. Proposition 4 observed that 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 results in a feasible solution since for such times t, $x_{j,t} < c \le 1/2$. This satisfies all constraints in $\mathsf{LP}_{\mathsf{heavy}}$ 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 ℓ_k norm of flow time.

Definition 6. *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 processed at all at time t.)*

We are now ready to introduce our key lemma, but, before we do, we need to define several notations. 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, $\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y')$ [$\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y')$, 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 7. *There exists a randomized polynomial time algorithm that constructs a feasible priority-preserving solution y^* to $\mathsf{LP}_{\mathsf{heavy}}$ such that*

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

given any feasible priority-preserving solution y to $\mathsf{LP}_{\mathsf{heavy}}$ as input.

By repeatedly applying Lemma 7, we will convert a fractional solution to $\mathsf{LP}_{\mathsf{heavy}}$ to an integral solution without increasing the cost too much. After taking the expectation on both sides over 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 and the fractional objective decrease geometrically each time the lemma is applied. To repeatedly apply the lemma, we ensure the invariant that the LP solution always remains priority-preserving and crucially use it for our analysis.

To apply Lemma 7 the first time, we need to have an initial solution to $\mathsf{LP}_{\mathsf{heavy}}$ that is priority-preserving. This is shown in the following lemma.

LEMMA 8. *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 $\mathsf{LP}_{\mathsf{heavy}}$. Proposition 5 ensures that this solution has the desired cost. The rest of the proof focuses on showing that it is priority-preserving.

Let t' be the *first* time that the LP is not priority-preserving. That is, it is the first time where there are two jobs i and j where $r_j \leq r_i$ and $p_j < p_i$ and it is the case that $y_{j,t'} > 0$, but $y_{i,t'} < 1$ and $y_{i,t'-1} = 1$. Let T' be the set of times $t \geq t'$ 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 α and increase $y_{i,t}$ by α. Note that this can only reduce the LP objective since we know that $r_j \leq r_i$, implying 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.[4] Hence we only need to show that the new LP solution y remains feasible.

We show that Constraints (4) remain satisfied. It is straightforward to see that the other constraints are satisfied. To see that Constraints (4) are satisfied, consider any time $t \in T'$. Consider any interval I such that $j \in H_I$ and $b(I) = t$. Since we decreased $y_{j,t}$ by α, the reader may wonder if Constraint (4) is violated. First note that that for i and j to be in H_I, it must be the case that j and i are not in $S_{c,I}$. This is true since both $y_{j,t}$ and $y_{i,t}$ are less than 1 and therefore less than c. Now 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 $r_i \leq t' \leq t = b(I)$, 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 is offset by the increase of $y_{i,t}$ in the left-hand-side of the constraint and the constraint remains satisfied. This is a contradiction to the fact that y is an optimal solution to $\mathsf{LP}_{\mathsf{heavy}}$.

[4] 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 Proposition 5. □

Using Lemma 7 and the previous lemma, we can find an integral solution to $\mathsf{LP}_{\mathsf{heavy}}$ with low cost.

COROLLARY 9. *There exists a polynomial time algorithm that produces w.h.p. 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 7 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 only need 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 (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. The number of iterations we need to apply Lemma 7 is polynomially bounded by the input size with high probability because the fractional LP cost decreases geometrically in expectation each application.

To upper bound the cost of the integral solution, 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 $c = \frac{1}{2^{k+4}}$. Combining this with Lemma 8, we obtain the claimed upper bound on the expected LP cost.

Formally, by applying Lemma 8 initially and Lemma 7 iteratively, we can construct a sequence of feasible solutions $y^0, y^1, ..., y^L$ to $\mathsf{LP}_{\mathsf{heavy}}$, where the final solution y^L is completely integral. By Lemma 8, we have,

$$\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y^0) \leq \mathsf{LP}_{\mathsf{heavy}}(y^0) \leq \frac{2}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}}) = O(1)^k \mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$$

By taking the expectation over y of both sides in the second claim of Lemma 7, for any $l \geq 1$, we have $\mathrm{E}[\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y^l)] \leq \frac{1}{2}\mathrm{E}[\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y^{l-1})]$. This immediately yields,

$$\mathrm{E}[\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y^l)] \leq \frac{1}{2^l}\mathrm{E}[\mathsf{LP}_{\mathsf{heavy}}(y^0)] = \frac{1}{2^l}\mathsf{LP}_{\mathsf{heavy}}(y^0)$$

Finally, from the first claim of Lemma 7 (after taking the expectation over y), we have,

$$\begin{aligned}
\mathrm{E}[\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y^L)] &= \mathrm{E}[\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y^0)] + \sum_{l=1}^{L} \left(\mathrm{E}[\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y^l)] - \mathrm{E}[\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y^{l-1})]\right) \\
&\leq \mathrm{E}[\mathsf{LP}_{\mathsf{heavy}}(y^0)] + \sum_{l=1}^{L} O(1)^k \cdot \mathrm{E}[\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y^{l-1})] \\
&\leq \mathsf{LP}_{\mathsf{heavy}}(y^0) + \sum_{l=1}^{L} O(1)^k \cdot \frac{1}{2^l}\mathsf{LP}_{\mathsf{heavy}}(y^0) \\
&= O(1)^k \mathsf{LP}_{\mathsf{heavy}}(y^0) = \frac{1}{c} O(1)^k \mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})
\end{aligned}$$

Thus, we have constructed a feasible integer solution to $\mathsf{LP}_{\mathsf{heavy}}$ whose objective is at most $\frac{1}{c}O(1)^k \mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$ in expectation, as desired. ∎

Thus, it only remains to prove Lemma 7.

3.1. Iterative Quasi-Uniform Sampling (Proof of Lemma 7). Our remaining goal is to construct a new LP solution y^* that satisfies the properties claimed in Lemma 7. To prove Lemma 7, fix a feasible priority-preserving solution y to $\mathsf{LP}_{\mathsf{heavy}}$, which we can assume exists inductively due to Lemma 8. We now describe the *randomized rounding* procedure. After each iteration of the randomized rounding, we will observe that the current fractional solution is priority-preserving.

First set $y^*_{j,t} = 1$ if $y_{j,t} = 1$ – this will ensure that the rounding is *'integrally' monotone*, meaning that if $y_{j,t} = 1$ at some point, it remains so throughout the iterations. Next, we use a randomized rounding which consists of the following two steps.

Step (i). Sampling Completion Times. For each job j and $i \geq 0$, let $\beta_{j,i} := 2^i + r_j$. We sample the completion time $\beta_{j,i}$ independently with probability $\min\{1, \frac{1}{c} \cdot y_{j,\beta_{j,i}}\}$ for each j and $i \geq 0$. 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 $y_{j,\beta_{j,i}}$, *exactly* the amount job j is fractionally incomplete at the time $\beta_{j,i}$. 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 $r_j \leq t \leq \beta_{j,i}$. If multiple completion times are sampled for a job, then the job's completion time is set to be the maximum of all completion times sampled for the job.

Step (ii). Constructing a Less Fractional LP Solution. After step (i), some interval constraints (Constraints (4)) may be satisfied. In particular, an interval $I = [a(I), b(I)] \in \mathcal{H}$ is satisfied if $y^*_{j,b(I)} = 1$ for some job j in H_I. For convenience, we remove such intervals from $\mathcal{H}$ and focus on satisfying constraints for the remaining intervals in $\mathcal{H}$. This is without loss of generality, as the randomized rounding is integrally monotone as observed above. *Additionally*, we remove every interval I from $\mathcal{H}$ if there is a job $j \in H_I$ such that $\beta_{j,i}$ was sampled in Step (i) and $\beta_{j,i+2} = r_i + 4 \cdot 2^i > b(I)$. Intuitively, quadrupling every job's flow time at the end of the rounding will satisfy such intervals.

Note that for any (remaining) interval $I \in \mathcal{H}$, we have $y_{j,b(I)} \leq c$ 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 $\sum_{j \in H_I \setminus E_I} y_{j,b(I)} \leq \frac{3}{5}$. For each (remaining) interval $I \in \mathcal{H}$ and for every job $j \in N_I$, we set $y^*_{j,b(I)} = \max\{y^*_{j,b(I)}, 2y_{j,b(I)}\}$.[5] Also to ensure that Constraints (3) are satisfied, we minimally increase $y^*_{j,t}$ if necessary for all $r_j \leq t \leq b(I)$ so that $y^*_{j,t} \geq 2y_{j,b(I)}$.

We repeat the above until $\mathcal{H}$ is non-empty. Observe that the size of $\mathcal{H}$ never increases over iterations. In particular, an interval removed from $\mathcal{H}$ will never be added back to $\mathcal{H}$. Once $\mathcal{H}$ is empty, we quadruple every job j's flow time: if t' is the largest time such that $y_{j,t'} = 1$, then let $y_{j,t} = 1$ for all $t \in (r_j, r_j + (t' - r_j) \cdot 4]$.

PROPOSITION 10. *At the end of the rounding algorithm, the constructed y is a feasible solution to $\mathsf{LP}_{\mathsf{heavy}}$ (with no Constraints (4) deleted).*

[5] This is well-defined because $c < 1/2$, and if $y_{j,b(I)} \geq c$, then we would have $y^*_{j,b(I)} = 1$.

Proof. When a heavy interval I is removed from $\mathcal{H}$, there must exist a job $j \in H_I$ and some i such that $y^*_{j,\beta_{j,i}} = 1$ (so, after renaming y^* by y, we have $y_{j,\beta_{j,i}} = 1$) and $\beta_{j,i+2} = r_i + 4 \cdot 2^i > b(I)$. Recall that the algorithm is integrally monotone, meaning that once $y_{j,t} = 1$ then it remains so until the end of the algorithm. Thus, thanks to the final step of quadrupling every job's flow time, y satisfies the constraint corresponding to I at the end. ∎

Since the last step only increases the LP objective by a factor of 4^k and our final goal is to find an integer schedule that is $O(1)^k$-approximate for minimizing the total kth power of flow time, we will henceforth ignore the step.

Intuition for the Randomized Rounding Algorithm. The iterative randomized rounding makes the solution more integral in each iteration. Here the procedure keeps the current solution feasible while decreasing the cost of the fractional part of the LP solution by a constant factor in each iteration. In Step (i), the algorithm samples a completion time for each job. For ease of analysis, the algorithm only distinguishes completion times for a job when they differ by more than a constant factor in length from the job's arrival. This is why we sample j's completion time to be one of the $\beta_{j,i}$ times for some i. Intuitively, by using the threshold rounding and 'boosted-up' randomized rounding, we 'oversample' the completion time of a job by up to a constant factor over the LP solution.

Step (ii) is concerned with intervals that are not satisfied by Step (i), and this is the more interesting step of the algorithm. 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. Each constraint is defined by a heavy interval I and a constraint can be satisfied by setting a job j's completion times to later than $b(I)$, the end of the interval for some job j that arrives during the interval I. To safely remove a job from the current LP solution (more precisely, remove fractional completion times for a job), we should 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 over the LP solution at the beginning of the iteration. 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 small probability, which will be more than enough to cancel the effect of increasing critical jobs contributions. Finally, we also remove additional intervals from $\mathcal{H}$ if quadrupling a job's flow time at the end would satisfy the corresponding constraint. This trick is to fix the issues occurring due to only sampling $\beta_{j,i}$ times which are time points that are power of two away from the job arrival times.

We begin the analysis by showing y^* is feasible and priority-preserving. First consider showing that it is priority-preserving.

LEMMA 11. *At the end of the Step (ii), the solution y^* is priority-preserving if y is priority-preserving.*

Proof. For the sake of contradiction, suppose y^* is not priority-preserving. Consider a pair of jobs j and j' that violates the priority-preservation property in y^*: $r_j \le r_{j'}$, $p_j < p_{j'}$, yet for some time $t \ge r_{j'}$, we have $y^*_{j,t} > 0$ and $y^*_{j',t} < 1$. In words, j' has been partially processed at time t while the job j of higher priority hasn't completed. Recalling that our rounding is integrally monotone, it must be the case that $y_{j',t} < 1$. Then, it must be the case that $y_{j,t} = 0$ due to y being priority-preserving. However, our rounding doesn't increase a variable if its value is

0, meaning that $y^*_{j,t} = 0$. This is a contradiction. ∎

Next we show the solution y^* is feasible.

PROPOSITION 12. *For all $I = [a(I), b(I)] \in \mathcal{H}$, we have $\sum_{j \in N_I} y_{j,b(I)} \geq \frac{1}{2}$.*

Proof. Jobs are added to the set E_I until $\sum_{j \in H_I \setminus E_I} y_{j,b(I)} \leq \frac{3}{5}$. Thus $\sum_{j \in E_I} y_{j,b(I)} \leq \frac{3}{5} + c \leq \frac{3}{5} + \frac{1}{10} \leq \frac{1}{2}$. Here we use the fact that for any job j in H_I, $y_{j,b(I)} \leq c \leq \frac{1}{10}$ which follows from the definition of c and the threshold rounding. The lemma follows from definition of N_I. ∎

The following lemma shows that $\mathsf{LP}_{\mathsf{heavy}}$ is satisfied. Note this this lemma only considers remaining intervals in $\mathcal{H}$ and every deleted interval is guaranteed to be satisfied at the end by quadrupling every job's flow time as observed in Proposition 10.

LEMMA 13. *At the end of the Step (ii), y^* satisfies Constraint (4) for every (remaining) interval $I \in \mathcal{H}$.*

Proof. It is easy to see that y^* satisfies Constraints (3) after both steps (i) and (ii). We only need to show that if Constraint (4) for an interval I is not satisfied by step (i), then it is fractionally satisfied by jobs in N_I. By Proposition 12, we have $\sum_{j \in N_I} y_{j,b(I)} \geq \frac{1}{2}$. Our algorithm sets $y^*_{j,b(I)} \geq 2y_{j,b(I)}$, thus satisfies Constraint (4) for I. ∎

Next we upper bound the cost of the new LP solution y^* we constructed following steps (i) and (ii). We first bound the cost for times set integrally by step (i). We will upper bound $\mathrm{E}[\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y^*)] - \mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y)$ and $\mathrm{E}[\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y^*)]$, respectively. The following inequality is useful for our analysis and follows from an elementary algebra. This inequality shows that the cost of completing a job j at time $\beta_{j,i}$ is at most 2^k larger than the cost of completing the job at time $\beta_{j,i-1}$.

PROPOSITION 14. *For all j and $i \geq 0$, $\Delta_{\beta_{j,i+1} - r_j} \leq 2^k \Delta_{\beta_{j,i} - r_j}$. Also Δ_l is increasing in l.*

The following is a restatement of the first inequality stated in Lemma 7 for easy reference.

LEMMA 15. $\mathrm{E}[\mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y^*)] - \mathsf{LP}^{\mathsf{int}}_{\mathsf{heavy}}(y) \leq \frac{2^{k+1}}{c} \mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y)$.

Proof. Fix a job j and an integer $i \geq 0$. Assume that $y_{j,\beta_{j,i}} < 1$ since otherwise in the current iteration, there is no increase in the integral cost for job j at times in $(r_j, \beta_{j,i}]$. Note that if $y_{j,t} < 1$, then we set the value of $y^*_{j,t}$ to 1 only in step (i). If time $\beta_{j,i}$ is sampled to be j's completion time in step (i), then $y^*_{j,t}$ is set to 1 for all $r_j < t \leq \beta_{j,i}$. The incurred cost is upper bounded by $\sum_{r_j < t \leq \beta_{j,i}} \Delta_{t - r_j} \leq \sum_{r_j < t \leq \beta_{j,i}} \Delta_{\beta_{j,i} - r_j} = (\beta_{j,i} - r_j) \Delta_{\beta_{j,i} - r_j} = 2^i \Delta_{\beta_{j,i} - r_j}$, where the first inequality follows from Proposition 14. The probability of $\beta_{j,i}$ being sampled is at most $\frac{1}{c} y_{j,\beta_{j,i}}$. Thus the expected increase in the integral LP cost due to $\beta_{j,i}$ being sampled as j's completion time is at most

$$\frac{1}{c} y_{j,\beta_{j,i}} 2^i \Delta_{\beta_{j,i} - r_j}. \tag{5}$$

On the other hand, because $y^*_{j,t}$ is set to 1 only in step (i) when $y_{j,t} < 1$, and j's completion times are sampled only at time $\beta_{j,i'}$, $i' \geq 0$, it follows that $y_{j,t'} < 1$ for all $t' \in (\beta_{j,i-1}, \beta_{j,i}]$. Thus j's fractional LP cost during $(\beta_{j,i-1}, \beta_{j,i}]$ is at least,

$$\sum_{t = \beta_{j,i-1} + 1}^{\beta_{j,i}} \Delta_{t - r_j} y_{j,t} \geq 2^{i-1} \Delta_{\beta_{j,i-1} - r_j} y_{j,\beta_{j,i}} \geq \frac{1}{2^{k+1}} 2^i \Delta_{\beta_{j,i} - r_j} y_{j,\beta_{j,i}}, \tag{6}$$

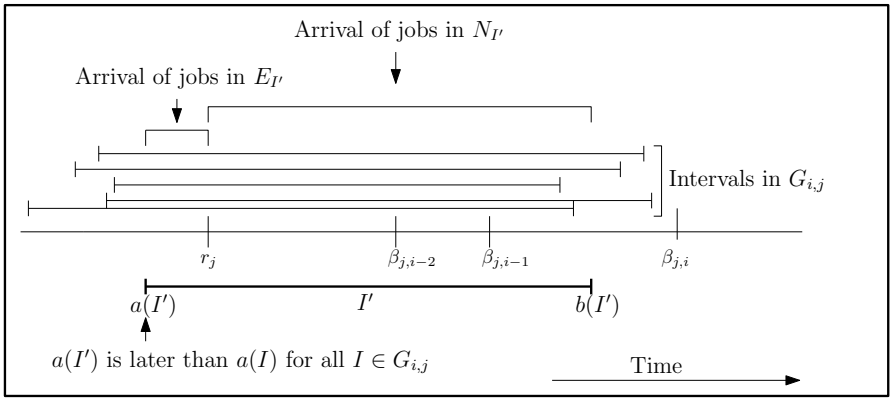

FIG. 2. *Intervals in $G_{i,j}$. Note that all jobs in $E_{I'}$ arrive during every interval in $G_{i,j}$. Due to this, if a job in $E_{I'}$ is given a completion time later than $\beta_{j,i}$ then all intervals in $G_{i,j}$ are satisfied.*

where the inequalities follow by Proposition 14 and Constraints (3). By charging Eqn. (5) to Eqn. (6), we have the lemma. ∎

To show that the fractional LP cost decreases significantly in each iteration, for each fixed job j, we would like to bound the probability that some interval for which j is critical is unsatisfied after step (i). 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 $j \in N_I$. Note that $I \in G_{i,j}$ only if $r_{j'} + 4(r'_{j'} - r_j) \leq b(I)$ where $r'_{j'}$ is the largest time t such that $y_{j',t} = 1$ since otherwise I would have been removed from $\mathcal{H}$ in the previous iterations.

LEMMA 16. *Fix any job j and any integer $i \geq 1$. The probability that there exists an interval in $I \in G_{i,j}$ that is not satisfied (therefore still in $\mathcal{H}$) after step (i) of the algorithm 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

$$\sum_{j' \in E_{I'}} y_{j',b(I')} \geq 2/5. \tag{7}$$

Further, by Lemma 8 and 11 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.

For a job $j' \in E_{I'}$ let $i_{j'}$ be the largest integer such that $\beta_{j',i_{j'}} = r_{j'} + 2^{i_{j'}}$ is smaller than or equal to $\beta_{j,i-1}$. 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 $\beta_{j',i_{j'}}$ or later by step (i) then *every* interval in $G_{i,j}$ is removed from $\mathcal{H}$. If we can show this and additionally show that some job $j' \in E_{I'}$ is given a completion time later than $\beta_{j',i_{j'}}$ 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.

We noted above that $p_{j'} \geq p_j$. Hence job j' can be used to satisfy Constraint (4) for I since job j is large enough 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 $\beta_{j',i_{j'}}$ by Step (i) then the constraint for I is removed from $\mathcal{H}$. 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 so that it is in H_I and can satisfy Constraint (4) for I. The algorithm removes all such constraints I where $r_{j'} + 4(\beta_{j',i_{j'}} - r_{j'}) \geq b(I)$. Notice that $r_{j'} + 4(\beta_{j',i_{j'}} - r_{j'}) \geq b(I)$ because $i_{j'} \geq i - 1$ since $r_{j'} < r_j$.

It only remains to bound the probability of the bad event that no job j' in $E_{I'}$ is given a deadline at $\beta_{j',i_{j'}}$ or later. We know that completion time $\beta_{j',i_{j'}}$ is sampled for job j' with probability $\frac{1}{c} y_{j',\beta_{j',i_{j'}}}$, which is greater than $\frac{1}{c} y_{j',b(I')}$ due to Constraint (3). Note that $y_{j',b(I')} \leq y_{j',i'_j} < c$. Otherwise, I would have been already removed by the above argument. Thus, the probability that no job j' in $E_{I'}$ has a completion time sampled at $\beta_{j',i_{j'}}$ or later is at most

$$\prod_{j' \in E_{I'}} (1 - y_{j',b(I')}/c) \leq \prod_{j' \in E_{I'}} \exp(-y_{j',b(I')}/c) \leq \exp(\sum_{j' \in E_{I'}} -y_{j',b(I')}/c) \leq \exp(-\frac{2}{5c}),$$

where the last inequality follows due to Eqn. (7). ∎

The previous lemma will allow us to bound the fractional LP cost in the new LP solution y^* as follows. Note that this is a restatement of the second inequality stated in Lemma 7 using that $\exp(-x)$ is approximately $1 - x$ when x is less than a small constant.

LEMMA 17. $\mathrm{E}[\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y^*)] \leq 2\exp(-(\frac{2}{5c}))\mathsf{LP}^{\mathsf{frac}}_{\mathsf{heavy}}(y)$.

Proof. For a fixed job j and $i \geq 1$, in the worst case $y_{j,t}$ can double at times $t \in (\beta_{j,i-1}, \beta_{j,i}]$, i.e. $y^*_{j,t} \leq 2y_{j,t}$. We know that this can happen only when some interval in $I \in G_{i,j}$ is not satisfied after step (i), which occurs with probability at most $\exp(-\frac{2}{5c})$ due to Lemma 16. Thus, we have the claimed inequality for j's fractional cost during $(\beta_{j,i-1}, \beta_{j,i}]$. Summing over all jobs and i, we have the lemma thanks to the linearity of expectation.

4. Light Intervals. In this section, an algorithm is defined that sets completion times for the jobs to satisfy all of the constraints for light intervals. Recall that after the threshold rounding, intervals that correspond to unsatisfied constraints are partitioned into two groups, $\mathcal{H}$ and $\mathcal{L}$. This section shows how to set job completion times to satisfy Constraints (2) for all light intervals, $\mathcal{L}$. The total cost of the completion times set in this section will be at most $\frac{O(1)^k}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$ in expectation. As mentioned, this rounding closely follows [7]. For completeness the rounding procedure along with the analysis is given in this section.

4.1. Preprocessing the LP Solution. The algorithm takes as input the solution $\tilde{x}$ to $\mathsf{LP}_{\mathsf{main}}$ after the the threshold rounding in Section 2. The algorithm begins by preprocessing the solution.

The input is a solution $\tilde{x}_{j,t}$ such that $\sum_{j \in L_I} p_j \tilde{x}_{j,b(I)} \geq \frac{1}{2}(V(I) - |I| - P(S_{c,I}))$ for all $I \in \mathcal{L}$. Further, for every $j \in L_I$, $p_j < V(I) - |I| - P(S_{c,I})$ and $\tilde{x}_{j,b(I)} < c$.

The algorithm begins by first increasing all $\tilde{x}_{j,t}$ where $\tilde{x}_{j,t} < c$ by a factor of 16. For each job j and $i \geq 0$, define $\beta_{j,i} := r_j + 2^i$; also define $\beta_{j,-1} = r_j$. For each j, $i \geq 1$ and all times $t \in (\beta_{j,i-1}, \beta_{j,i}]$, set $\tilde{x}_{j,t} = \tilde{x}_{j,\beta_{j,i-1}}$. This only can increase the value of a variable $\tilde{x}_{j,t}$ because $\tilde{x}$ satisfies Constraints (1). All Constraints (1) remain satisfied: Those in Constraints (2) remain satisfied since variables only increase and the remaining constraints are trivially satisfied. Further, using the fact that $(\beta_{j,i+1} - r_j)^k \leq 2^k(\beta_{j,i} - r_j)^k$, it can be established that the LP cost only slightly increases. These facts are shown in the following lemma.

LEMMA 18. *After the above rounding, the LP solution remains feasible and the cost increases by a factor of at most* 2^{3k+4}.

Proof. The proof focuses on the second claim as the first claim has been already proved above. The first step increases the objective by a factor of at most 16. We bound the change in the objective by considering each job j's contribution to the objective individually. Before the second step, the j's contribution to the objective is at least

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

Similarly, one can see that the objective increases to at most the following in the second step.

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

Using the fact that $\Delta_{\beta_{j,i+1}-r_j} \leq 4^k\Delta_{\beta_{j,i-1}-r_j}$ and including in the factor 16 increase due to the first step, the lemma follows. ∎

Let $D_I = (V(I) - |I| - P(S_{c,I}))$ be the residual demand for interval I. After the above modification the fractional solution $\tilde{x}$ ensures that for every light interval $I \in \mathcal{L}$, $\sum_{j \in L_I} p_j \tilde{x}_{j,b(I)} \geq 8D_I$ because if $j \in L_I$ then $\tilde{x}_{j,b(I)}$ increased by a factor 16. Further, it was initially the case that $\sum_{j \in L_I} p_j \tilde{x}_{j,b(I)} \geq \frac{1}{2}(V(I) - |I| - P(S_{c,I}))$ before the modification.

For the remainder of the analysis, for ease of analysis, it is assumed that every job has a size equal to a power of two. This can be done by rounding up each job size to the nearest power of two. Due to this modification of the job sizes, the analysis will guarantee that a solution is found that satisfies Constraint (2) for each interval

I in $\mathcal{L}$ with the demand doubled to $2D_I$. This will ensure that when job sizes are scaled back to their original size, the constraints are still satisfied.

It will be convenient to simplify the LP, similarly as was done for the case of heavy intervals. The following LP uses variables z. Let $z_{j,i}$ denote whether job j is completed at time $r_j + 2^i$. Let $i_0(j)$ be the smallest i such that $\tilde{x}_{j,\beta_{j,i}} < 1$. Let $i(j,I)$ denote the smallest i' such that $\beta_{j,i'} \geq b(I)$ for any interval I.

The new LP is the following. The variables z act like the variables x, except $z_{j,i}$ is index by jobs and times $\beta_{j,i}$ and $x_{j,t}$ is indexed by jobs and all times. The z variables only consider restricted times because $\tilde{x}_{j,t}$ has the same value for all $t \in (\beta_{j,i}, \beta_{j,i+1}]$ by definition of the modification to $\tilde{x}$. It is assumed that all jobs have size equal to a power to two and for notational convenience set there is a variable $z_{j,i_0(j)-1}$.

$$(\mathsf{LP}_{\mathsf{light}}) \qquad \min \sum_j \sum_{i \geq i_0(j)} \Delta'_{j,i} z_{j,i}$$

$$(8) \qquad s.t. \quad \sum_{j: r_j \in I, p_j < D_I} p_j \cdot z_{j,i(j,I)} \geq 2D_I \qquad \forall I \in \mathcal{L}$$

$$(9) \qquad z_{j,i} \leq z_{j,i-1} \qquad \forall j, i \geq i_0(j)$$

$$(10) \qquad z_{j,i_0(j)-1} = 1 \qquad \forall j$$

$$z_{j,i} \geq 0 \qquad \forall j, i \geq i_0(j),$$

In the LP, $\Delta'_{j,i} = (\beta_{j,i} - r_j)^k - (\beta_{j,i-1} - r_j)^k$. Set $\tilde{z}_{j,i} = \tilde{x}_{j,\beta_{j,i}}$. This solution is feasible and has cost at most $\frac{O(1)^k}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$ (see Proposition 4 and Lemma 18). Further, it is the case that for all intervals $I \in \mathcal{L}$,

$$(11) \qquad \sum_{j: r_j \in I, p_j < D_I} p_j \tilde{z}_{j,i(j,I)} \geq 8D_I.$$

Notice that in the summation of Constraint (8), we omit the condition $j \notin S_{c,I}$. Recall that $j \in L_I$ if and only if j arrives during I, $p_j < D_j$, and $j \notin S_{c,I}$. However, no job j in the summation is in $S_{c,I}$. This is because $j \in S_{c,I}$ only when $\tilde{x}_{j,\beta_{j,i(j,I)}} = 1$, and there is no variable $z_{j,i(j,I)}$ since $i(j,I) < i_0(j)$ by definition of $i_0(j)$.

Let $\mathsf{LP}_{\mathsf{light}}(\tilde{z})$ denote the objective of $\mathsf{LP}_{\mathsf{light}}$ for the solution $\tilde{z}$. In the following section, an integral solution $\bar{z}_{i,j}$ will be constructed to $\mathsf{LP}_{\mathsf{light}}$ whose cost is bounded by $\mathsf{LP}_{\mathsf{light}}(\tilde{z})$. This will be sufficient to derive a solution to $\mathsf{LP}_{\mathsf{main}}$ satisfying all of the light constraints. To see this, 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 kth power of flow time will be $(\beta_{j,i'} - r_j)^k$. Part of this job's cost, its kth power flow time up to $i_0(j)$, was already bounded in the analysis of the threshold rounding in Section 2 and Lemma 18. The remaining part of the job's cost, $(\beta_{j,i'} - r_j)^k - (\beta_{j,i_0(j)} - r_j)^k$, is exactly the contribution of job j to the objective of $\mathsf{LP}_{\mathsf{light}}$ under the solution $\bar{z}$. With the above arguments in place, 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 at most $\frac{O(1)^k}{c}\mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$ that satisfies all constraints for light intervals.

4.2. Reduction to a Geometric Covering Problem. The analysis now makes a connection between $\mathsf{LP}_{\mathsf{light}}$ and the following geometric covering problem in the plane. In this problem, there is a collection of axis-parallel aligned rectangles.

Denote $[x'_1, x'_2] \times [y'_1, y'_2]$ as a rectangle with $(x'_1, y'_1), (x'_1, y'_2), (x'_2, y'_1)$ and (x'_2, y'_2) as its four corners.

The R2M Problem. The input is a collection $\mathcal{Q}$ of points q in 2D Euclidean space, each with an integer demand D_q, and a collection of axis-parallel rectangles B of the form $[0, x_B] \times [y_B^1, y_B^2]$, 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.

It will be established that finding an integral solution to $\mathsf{LP}_{\mathsf{light}}$ can be reduced to the R2M problem. To do so, $\log_2 \max_j p_j$ instances of the R2M problem will be created. Index an instance by $\ell \in \{0, 1, 2, \ldots, \log_2 \max_j p_j\}$. Each instance $\mathcal{I}_\ell$ has rectangles corresponding to jobs of size 2^ℓ.

Create the instance $\mathcal{I}_\ell$ 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 be defined shortly. For each job j of size 2^ℓ, create a collection of rectangles $\{B_{j,i}\}_{i \geq i_0(j)}$ where $B_{j,i}$ denotes a rectangle, $[0, r_j] \times [\beta_{j,i-1} + 1, \beta_{j,i}]$, of cost $\Delta'_{j,i} := (\beta_{j,i} - r_j)^k - (\beta_{j,i-1} - r_j)^k$. Set the demand of interval I to be $D_{I,\ell} = \lfloor \sum_{j,i: p_j = 2^\ell, q_I \in B_{j,i}} \tilde{z}_{j,i} \rfloor$. This completes the description of R2M instances, $\{\mathcal{I}_\ell\}$.

It was shown that the R2M problem has a randomized polynomial time constant approximation algorithm [7]. Further, the approximation is based on rounding a solution to 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.

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

$$(12) \qquad s.t \sum_{j,i: q_I \in B_{j,i}, p_j = 2^\ell} z'_{j,i} \geq D_{I,\ell} \qquad \forall I \in \mathcal{L}$$

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

From this IP, a LP relaxation LP_ℓ is obtained by allowing $z'_{j,i} \in [0, 1]$. Observe that $\tilde{z}_{j,i}$ is a feasible LP solution to this LP_ℓ. This is because $q_I \in B_{j,i}$ if and only if $a(I) \leq r_j$ and $\beta_{j,i-1} < b(I) \leq \beta_{j,i}$. By definition of $D_{I,\ell}$, it is the case that $D_{I,\ell} \leq \sum_{j,i: p_j = 2^\ell, q_I \in B_{j,i}} \tilde{z}_{j,i}$, which implies that $\tilde{z}_{j,i}$ satisfies Constraints (12).

The algorithm in [7] rounds any feasible solution to LP_ℓ to an integral solution and ensures the objective increases by at most a $O(1)$ factor. Let $\{\bar{z}'_{j,i}\}_{j: p_j = 2^\ell}$ be the integral solution to LP_ℓ obtained using the constant approximation from [7] based on rounding the solution $\tilde{z}_{j,i}$ to LP_ℓ. The rounding ensures that $\mathsf{LP}_\ell(\bar{z}') = O(1)\mathsf{LP}_\ell(\tilde{z})$.

To construct an integral solution $\bar{z}$ to $\mathsf{LP}_{\mathsf{light}}$ initially set $\bar{z}_{j,i} = 0$. Then if $\bar{z}'_{j,i} = 1$, set $\bar{z}_{j,i_0(j)} = \bar{z}_{j,i_0(j)+1} = ... = \bar{z}_{j,i} = 1$. In the remaining proof, it is established that the cost of $\bar{z}$ is bounded and this is a feasible solution to $\mathsf{LP}_{\mathsf{light}}$.

LEMMA 19. $\mathsf{LP}_{\mathsf{light}}(\bar{z}) = \frac{O(1)}{c} \mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$.

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 LP_ℓ are only defined over jobs of size 2^ℓ, we have $\mathsf{LP}_{\mathsf{light}}(\bar{z}) = \sum_\ell \mathsf{LP}_\ell(\bar{z}) = O(1) \sum_\ell \mathsf{LP}_\ell(\bar{z}') = O(1) \sum_\ell \mathsf{LP}_\ell(\tilde{z}) = O(1)\mathsf{LP}_{\mathsf{light}}(\tilde{z}) = \frac{O(1)}{c} \mathrm{OPT}(\mathsf{LP}_{\mathsf{main}})$. ∎

Finally, we show that $\bar{z}$ is a feasible solution to $\mathsf{LP}_{\mathsf{light}}$. It is easy to see that $\bar{z}$ satisfies Constraints (9) and (10). It only remains to show that $\bar{z}$ satisfies Constraint

(8) for each interval $I \in \mathcal{L}$.

$$\begin{aligned}
&\sum_{j:r_j \in I, p_j < D_I} p_j \cdot \bar{z}_{j,i(j,I)} \\
&\geq \sum_{\ell: 2^\ell < D_I} 2^\ell \sum_{j:r_j \in I, p_j = 2^\ell} \bar{z}'_{j,i(j,I)} \\
&= \sum_{\ell: 2^\ell < D_I} 2^\ell \sum_{i,j: q_I \in B_{j,i}, p_j = 2^\ell} \bar{z}'_{j,i} \\
&\geq \sum_{\ell: 2^\ell < D_I} 2^\ell D_{I,\ell} && [\bar{z}' \text{ is feasible to } \mathsf{LP}_\ell] \\
&= \sum_{\ell: 2^\ell < D_I} 2^\ell \lfloor \sum_{j,i: q_I \in B_{j,i}, p_j = 2^\ell} \tilde{z}_{j,i} \rfloor \\
&\geq \sum_{\ell: 2^\ell < D_I} 2^\ell \sum_{j,i: q_I \in B_{i,j}, p_j = 2^\ell} \tilde{z}_{j,i} - \sum_{\ell: 2^\ell < D_I} 2^\ell \\
&\geq \sum_{j:r_j \in I, p_j < D_I} p_j \tilde{z}_{j,i(j,I)} - 2D_I \geq 6D_I && [\text{Due to (11)}],
\end{aligned}$$

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 $\beta_{j,i'-1} < b(I) \leq \beta_{j,i'}$. To summarize, we have obtained a feasible integral solution $\bar{z}$ to $\mathsf{LP}_{\mathsf{light}}$ and bounded its cost by the optimal objective of $\mathsf{LP}_{\mathsf{main}}$ in Lemma 19.

5. When Jobs' Sizes or Release Times are Not Polynomially Bounded by n. In this section, we reduce, losing only $1+\epsilon$ factor in the approximation ratio, any instance to one where all job sizes and arrival times are polynomially bounded by n, for which we already showed a $O(1)$-approximation.

Let $p_{\max} = \max_j p_j$. Let $\epsilon > 0$ be an arbitrarily small constant such that $1/\epsilon$ is an integer. We first consider jobs' arrival times. We say that two jobs are adjacent if no other jobs arrive between the two. We observe that we can assume w.l.o.g. that no two adjacent jobs j and j' have arrival times that differ by more than $np_{\max}$. This is because no preemptive scheduler has incentives to idle the machine when there are jobs ready for processing. Therefore, if there exist such a pair of jobs j and j', $r_j < r_{j'}$, all jobs arriving before j' complete before j' arrives, which implies that the instance can be decomposed into jobs arriving before j' and those arriving no later than j'. This observation allows us to assume w.l.o.g. that no two jobs have arrival times that differ by more than $n^2 p_{\max}$.

For the remainder of the section it is assumed w.l.o.g. that $\min_j r_j = 0$ and $\max_j r_j$ is polynomially bounded by $p_{\max}$ and n. 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 job arrival times and sizes become integer multiples of δ. More precisely, for each job j, let $\bar{p}_j$ be an integer multiple of δ such that $p_j - 3\delta < \bar{p}_j \leq p_j - 2\delta$; if $p_j \leq 3\delta$, then $\bar{p}_j = 0$. Also let $\bar{r}_j$ be an integer multiple of δ such that $r_j \leq \bar{r}_j < r_j + \delta$. Let $\bar{\mathcal{I}}$ denote the problem instance where $\bar{r}_j$ is the arrival of job j and $\bar{p}_j$ is the processing time of job j.

LEMMA 20. *Consider an optimal schedule σ^* for $\mathcal{I}$ where each job j completes at time C^*_j. There exists a schedule $\bar{\sigma}^*$ for $\bar{\mathcal{I}}$ where (i) each job j is processed by $\bar{p}_j$ units during $[\bar{r}_j, \bar{C}^*_j]$, where $\bar{C}^*_j \leq C^*_j$ is an integer multiple of δ, and (ii) at most one job is processed during each interval in $\mathcal{R} = \{[i\delta, (i+1)\delta] : i \text{ is an integer}\}$.*

Proof. We construct $\bar{\sigma}^*$ for $\bar{\mathcal{I}}$, from σ^* for $\mathcal{I}$, where each job completes earlier than in schedule σ^*. To see this, notice that one can process job j during time intervals $[r_j, \bar{r}_j]$ and $[\bar{C}^*_j := \lfloor C^*_j/\delta \rfloor \cdot \delta, C^*_j]$ by at most 2δ units. Job j's size in $\bar{\mathcal{I}}$ is smaller than in $\mathcal{I}$ by at least 2δ; we may update $\bar{C}^*_j$ later. If $\bar{\sigma}$ is created to match σ, except that it ignores the work that was done for job j during $[r_j, \bar{r}_j]$ and $[\bar{C}^*_j, C^*_j]$ in σ^*, each job j is processed completely or by more than needed during $[\bar{r}_j, \bar{C}^*_j]$. By dropping unnecessary processing, we make $\bar{\sigma}^*$ feasible.

We now interpret the schedule $\bar{\sigma}^*$ as a fractional flow between jobs and intervals in $\mathcal{R}$. More precisely, consider a bipartite graph between jobs and intervals in $\mathcal{R}$. If job j is processed during an interval $V \in \mathcal{R}$, then there is an edge from the job to the interval and the edge sends a flow of value equal to the number of time units of processing done on j during V, divided by δ. Thus, each job j sends to intervals in $\mathcal{R}$ a total of flow of value $\bar{p}_j/\delta$, which is an integer, and each interval receives a flow of value at most 1. Because of the integrality of network flow, it means that there is an integral flow where each interval V is matched with at most one job. If an interval $V \in \mathcal{R}$ is matched with a job j, then the whole interval V is dedicated to j's processing. In this schedule, because of the way the graph is constructed, j's completion time can be only smaller than $\lfloor C^*_j/\delta \rfloor \cdot \delta$ and is an integer multiple of δ. We set $\bar{C}^*_j$ to be j's completion time. This schedule satisfies all the desired properties. ∎

Lemma 20 shows that there exists a schedule $\bar{\sigma}^*$ for $\bar{\mathcal{I}}$ where each job j is processed alone during each interval or not at all, and completes no later than in the optimal schedule for the original instance $\mathcal{I}$. Since every job has a size $\bar{p}_j$ that is an integer multiple of δ in the modified instance $\bar{\mathcal{I}}$, and every interval in $\mathcal{R}$ has a length that is an integer multiple of δ, this means that in the schedule $\bar{\sigma}^*$, there are exactly $\bar{p}_j/\delta$ intervals in $\mathcal{R}$ dedicated to processing each job j. This allows us to scale down all parameters of $\bar{\mathcal{I}}$, jobs arrival time and sizes, uniformly by a factor of δ. Note that all parameters in the resulting instance are polynomially bounded by n, and we have already shown a $O(1)$-approximation for such instances. Using our $O(1)$-approximation algorithm, we can obtain a schedule for the scaled-down instance $\bar{\mathcal{I}}$ and scale back it. Let $\bar{C}_j$ denote j's completion time in the resulting schedule $\bar{\sigma}$.

Corollary 21. $(\sum_j (\bar{C}_j - \bar{r}_j)^k)^{1/k} \leq O(1) \cdot (\sum_j (\bar{C}^*_j - \bar{r}_j)^k)^{1/k} \leq O(1) \cdot (\sum_j (C^*_j - r_j)^k)^{1/k}$.

Proof. The first inequality follows, as we obtained $\bar{\sigma}$ for $\bar{\mathcal{I}}$ using our $O(1)$-approximation. The second inequality is due to Lemma 20 and the fact that $r_j \leq \bar{r}_j$ for all j. ∎

Finally, we convert the schedule $\bar{\sigma}$ for $\bar{\mathcal{I}}$ into our final schedule σ for $\mathcal{I}$ without increasing the objective too much.

Lemma 22. *There exists a feasible schedule σ for $\mathcal{I}$ where each job j's completion time C_j is at most $\bar{C}_j + 3n\delta$. Further, such a schedule can be found in polynomial time.*

Proof. Note that $\bar{\sigma}$ becomes a feasible schedule for $\mathcal{I}$ if each job j is processed by $p_j - \bar{p}_j$ more units, as it starts getting processed no earlier than $\bar{r}_j \geq r_j$ and is processed by $\bar{p}_j$ units in $\bar{\sigma}$. For notational simplicity, assume that job are ordered in increasing order of $\bar{C}_j$. Then, by adding extra processing to jobs in this order while pushing back later jobs, 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, which immediately yields the lemma since $p_j - \bar{p}_j \leq 3\delta$. ∎

LEMMA 23. *Let F_j denote j's flow time in $\mathcal{I}$, and let $\bar{F}_j$ denote the analogous quantity for instance $\bar{\mathcal{I}}$. Then, we have $F_j \leq \max\{(1+4\epsilon)\bar{F}_j, 5\epsilon\frac{p_{\max}}{n}\}$.*

Proof. To compare F_j to $\bar{F}_j$, we consider two cases. The first case is when $\bar{F}_j \geq \frac{\epsilon p_{\max}}{n}$. In this case, $F_j = C_j - r_j \leq \bar{C}_j + 3n\delta - \bar{r}_j + \delta \leq \bar{F}_j + 4n\delta \leq (1+4\epsilon)\bar{F}_j$. Otherwise, $F_j \leq \bar{F}_j + 4n\delta \leq \epsilon\frac{p_{\max}}{n} + 4\epsilon^2\frac{p_{\max}}{n} \leq 5\epsilon\frac{p_{\max}}{n}$. The lemma follows by taking the maximum of these two upper bounds. ∎

By summing over all jobs, we have

$$\sum_j F_j^k \leq \left((1+4\epsilon)^k \sum_j \bar{F}_j^k\right) + \frac{(5\epsilon)^k}{n^k} p_{\max}^k \cdot n \leq (1+9\epsilon)^k \sum_j \bar{F}_j^k$$

The last inequality follows from the fact that the kth power of flow time is at least $p_{\max}^k$ for any schedule. This inequality, together with Corollary 21, implies that our final schedule σ is an $O(1)$-approximation for the original instance $\mathcal{I}$, as desired.

6. Conclusion. This paper introduced the first constant approximation for the ℓ_k-norms of flow time on a single machine. After this work, independently the work of Barta, Garg and Kumar [8] and Feige, Kulkarni and Li [21] have shown an alternative algorithm for obtaining a constant approximation for the same problem. Their work further gave the first constant approximation for weighted flow time, resolving a major open problem in the area. The clear open questions are to determine best possible approximations for these problems.

It is additionally of interest to determine the best approximation ratio possible for the ℓ_k norms of flow time on identical machines. Currently, the results are known only with speed augmentation or when jobs arrive at time 0. It is worth mentioning that the work of Moseley [31] has used similar techniques as this paper for scheduling in the identical machines environment for generalizations of the ℓ_k norms of flow time.

REFERENCES

[1] F. N. Afrati, E. Bampis, C. Chekuri, D. R. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko, *Approximation schemes for minimizing average weighted completion time with release dates*, in IEEE Foundations of Computer Science, FOCS, 1999, pp. 32–44.

[2] S. Albers, *On randomized online scheduling*, in ACM Symposium on Theory of Computing, STOC, 2002, pp. 134–143.

[3] S. Anand, N. Garg, and A. Kumar, *Resource augmentation for weighted flow-time explained by dual fitting*, in ACM-SIAM Symposium on Discrete Algorithms, SODA, 2012, pp. 1228–1241.

[4] N. Bansal and K. Dhamdhere, *Minimizing weighted flow time*, ACM Trans. Algorithms, 3 (2007).

[5] N. Bansal and J. Kulkarni, *Minimizing flow-time on unrelated machines*, in ACM Symposium on Theory of Computing, STOC, 2015, pp. 851–860.

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

[7] N. Bansal and K. Pruhs, *The geometry of scheduling*, SIAM J. Comput., 43 (2014), pp. 1684–1698.

[8] J. Batra, N. Garg, and A. Kumar, *Constant factor approximation algorithm for weighted flow time on a single machine in pseudo-polynomial time*, CoRR, abs/1802.07439 (2018 (to appear in FOCS 2018)).

[9] C. Bussema and E. Torng, *Greedy multiprocessor server scheduling*, Oper. Res. Lett., 34 (2006), pp. 451–458.

[10] R. D. Carr, L. Fleischer, V. J. Leung, and C. A. Phillips, *Strengthening integrality gaps for capacitated network design and covering problems*, in ACM-SIAM Symposium on Discrete Algorithms, SODA, 2000, pp. 106–115.

[11] J. S. Chadha, N. Garg, A. Kumar, and V. N. Muralidhara, *A competitive algorithm for minimizing weighted flow time on unrelated machines with speed augmentation*, in ACM Symposium on Theory of Computing, STOC, 2009, pp. 679–684.

[12] D. Chakrabarty, E. Grant, and J. Könemann, *On column-restricted and priority covering integer programs*, in Integer Programming and Combinatorial Optimization, 14th International Conference, IPCO 2010, Lausanne, Switzerland, June 9-11, 2010. Proceedings, 2010, pp. 355–368.

[13] T. M. Chan, E. Grant, J. Könemann, and M. Sharpe, *Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling*, in ACM-SIAM Symposium on Discrete Algorithms, SODA, 2012, pp. 1576–1585.

[14] C. Chekuri, A. Goel, S. Khanna, and A. Kumar, *Multi-processor scheduling to minimize flow time with epsilon resource augmentation*, in ACM Symposium on Theory of Computing, STOC, 2004, pp. 363–372.

[15] C. Chekuri, S. Im, and B. Moseley, *Longest wait first for broadcast scheduling [extended abstract]*, in Workshop on Approximation and Online Algorithms, WAOA, 2009, pp. 62–74.

[16] C. Chekuri and S. Khanna, *A PTAS for minimizing weighted completion time on uniformly related machines*, in Automata, Languages and Programming, International Colloquium, ICALP, 2001, pp. 848–861.

[17] C. Chekuri and S. Khanna, *Approximation schemes for preemptive weighted flow time*, in ACM Symposium on Theory of Computing, STOC, 2002, pp. 297–305.

[18] C. Chekuri, R. Motwani, B. Natarajan, and C. Stein, *Approximation techniques for average completion time scheduling*, SIAM J. Comput., 31 (2001), pp. 146–166.

[19] J. Edmonds, S. Im, and B. Moseley, *Online scalable scheduling for the ℓ_k-norms of flow time without conservation of work*, in ACM-SIAM Symposium on Discrete Algorithms, SODA, 2011, pp. 109–119.

[20] J. Edmonds and K. Pruhs, *Scalably scheduling processes with arbitrary speedup curves*, ACM Transactions on Algorithms, 8 (2012), p. 28.

[21] U. Feige, J. Kulkarni, and S. Li, *A polynomial time constant approximation for minimizing total weighted flow-time*, in Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019, 2019, pp. 1585–1595.

[22] K. Fox and B. Moseley, *Online scheduling on identical machines using srpt*, in ACM-SIAM Symposium on Discrete Algorithms, SODA, 2011, pp. 120–128.

[23] N. Garg and A. Kumar, *Minimizing average flow time on related machines*, in ACM Symposium on Theory of Computing, STOC, 2006, pp. 730–738.

[24] A. Gupta, S. Im, R. Krishnaswamy, B. Moseley, and K. Pruhs, *Scheduling jobs with varying parallelizability to reduce variance*, in ACM Symposium on Parallelism in Algorithms and Architectures, SPAA, 2010, pp. 11–20.

[25] A. Gupta, R. Krishnaswamy, and K. Pruhs, *Online primal-dual for non-linear optimization with applications to speed scaling*, in Workshop on Approximation and Online Algorithms, WAOA, 2012.

[26] D. S. Hochbaum, ed., *Approximation Algorithms for NP-hard Problems*, PWS Publishing Co., Boston, MA, USA, 1997.

[27] S. Im and B. Moseley, *Online scalable algorithm for minimizing ℓ_k-norms of weighted flow time on unrelated machines*, in ACM-SIAM Symposium on Discrete Algorithms, SODA, 2011, pp. 95–108.

[28] S. Im and B. Moseley, *Fair scheduling via iterative quasi-uniform sampling*, in Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19, 2017, pp. 2601–2615.

[29] 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, (1982).

[30] J. Leung, L. Kelly, and J. H. Anderson, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, CRC Press, Inc., Boca Raton, FL, USA, 2004.

[31] B. Moseley, *Scheduling to approximate minimization objectives on identical machines*, in 46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece., 2019, pp. 86:1–86:14.

[32] B. Moseley, K. Pruhs, and C. Stein, *The complexity of scheduling for p-norms of flow and stretch - (extended abstract)*, in Integer Programming and Combinatorial Optimization , IPCO, 2013, pp. 278–289.

[33] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Springer Publishing Company,

Incorporated, 3rd ed., 2008.
[34] A. S. Tanenbaum, *Modern Operating Systems*, Prentice Hall Press, Upper Saddle River, NJ, USA, 2007.
[35] K. R. Varadarajan, *Weighted geometric set cover via quasi-uniform sampling*, in ACM Symposium on Theory of Computing, STOC, 2010, pp. 641–648.

Appendix A. Omitted Proofs.

Proof of Lemma 3. 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, the latest time t where $x_{j,t} = 1$. Note that $x_{j,r_j} = x_{j,r_j+1} = ... = x_{j,d_j} = 1$ and $x_{j,t} = 0$ for all $t > d_j$.

For the sake of contradiction, let j be the first job that is not completed by its deadline d_j by the EDF algorithm. 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 no later than d_j; we 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. This is because EDF would be scheduling this job or a job with earlier deadline at time $t_1 - 1$.

Consider Constraint (2) with $I = [t_1, d_j]$ and S being the subset of jobs that arrive during $[t_1, d_j]$ but have deadline later than d_j. Note that $V(I) - P(S)$ is the total size of jobs in $R(I) \setminus S$, those that arrive during $[t_1, d_j]$ and have deadline at or earlier than time d_j. Since EDF was always busy processing jobs in $R(I) \setminus S$ during $[t_1, d_j]$ but couldn't finish them by time d_j, it must be the case that $V(I) - P(S) > |I|$, which makes the right-hand-side of (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 in the given LP solution x. Hence the left-hand-side of (2) is 0, which is a contradiction to the constraint being satisfied. Thus, all jobs are completed by their respective deadlines by EDF. ∎