Geometry of Scheduling on Multiple Machines

Nikhil Bansal[*]and Jatin Batra[†]

Abstract

We consider the following general scheduling problem: there are m identical machines and n jobs all released at time 0. Each job j has a processing time p_j, and an arbitrary non-decreasing function f_j that specifies the cost incurred for j, for each possible completion time. The goal is to find a preemptive migratory schedule of minimum cost. This models several natural objectives such as weighted norm of completion time, weighted tardiness and much more.

We give the first $O(1)$ approximation algorithm for this problem, improving upon the $O(\log\log nP)$ bound due to Moseley (2019). To do this, we first view the job-cover inequalities of Moseley geometrically, to reduce the problem to that of covering demands on a line by rectangular and triangular capacity profiles. Due to the non-uniform capacities of triangles, directly using quasi-uniform sampling loses a $O(\log\log P)$ factor, so a second idea is to adapt it to our setting to only lose an $O(1)$ factor. Our ideas for covering points with non-uniform capacity profiles (which have not been studied before) may be of independent interest.

1 Introduction

The generalized scheduling problem gives a broad approach to model various min-sum scheduling objectives, and deals with the following setting. There are n jobs, with each job $j \in [n]$ having an integer release time r_j, which is the earliest time it can be processed, and an integer processing requirement p_j. In addition, for each job j there is an arbitrary non-decreasing non-negative cost function $f_j : \mathbb{Z} \to \mathbb{Z}$, that associates a cost for when j completes. The goal is to find a schedule that minimizes $\sum_j f_j(c_j)$, where c_j is completion time of job j. As f_j can be completely arbitrary for each j, this models several well-studied objectives and much more, see e.g. [2, 14, 15].

Single Machine. For the single machine case, Bansal and Pruhs [2] gave an $O(\log\log P)$ approximation for the problem, where P is the maximum to minimum job size ratio. To do this, they view the problem as a capacitated geometric set cover problem, and used LP-based techniques for geometric set cover problems with low union complexity, and knapsack-cover inequalities for capacitated problems. They also gave an $O(1)$ approximation when all the jobs have identical release times.

In exciting subsequent works, these results have been improved and built upon in various ways. For identical release times, [10] gave an improved $4+\varepsilon$ polynomial time approximation, and [1] gave a quasi-polynomial time approximation scheme. For general release times, better $O(1)$ approximation

[*]CWI and TU Eindhoven, the Netherlands. `bansal@gmail.com`
[†]CWI, the Netherlands. `Jatin.Batra@cwi.nl`

guarantees were obtained for various important objective functions such as ℓ_k norm of flow times [13] and weighted flow times [5, 11]. The general scheduling problem has also been considered in the online setting [14].

Multiple Machines. Despite remarkable progress on the single machine case, the multiple machine case is much less understood and seems significantly harder. A key difficulty is that the geometric reduction in [2] does not work for multiple machines.

Recently, Moseley [15] developed a surprising and elegant new approach for identical machines, with preemption and migration[1], in the case where all the jobs have the same release times. He considers a time-indexed LP formulation strengthened by certain job-cover and knapsack-cover inequalities. Using various structural properties of the LP, he applies the quasi-uniform sampling technique [16, 9, 13] in a clever way to round this LP and obtain an $O(\log\log nP)$ approximation.

1.1 Our Results and Techniques

Here we focus on the general scheduling problem studied by Moseley [15]: where jobs have the same release times and must be scheduled on multiple identical machines with preemption and migration. We will refer to this as GSP henceforth.

Our main result is the following, which improves on the $O(\log\log nP)$ approximation due to Moseley [15].

Theorem 1.1. *There is an $O(1)$ approximation algorithm for GSP.*

The result is based on two main ideas.

1.1.1 Reduction to a geometric covering problem

We first show that GSP reduces, up to $O(1)$ factors, to a clean capacitated geometric problem, of covering demands of points on a line by certain rectangular and triangular capacity profiles (see Figure 1). More formally, consider the following problem.

Definition 1.2 (The *TRC* problem)**.** *We are given a set of integer points P on a line, with point $p \in P$ having an integer demand d_p. There is a set Z of "capacity profiles", where a profile $z \in Z$ has cost w_z with either*

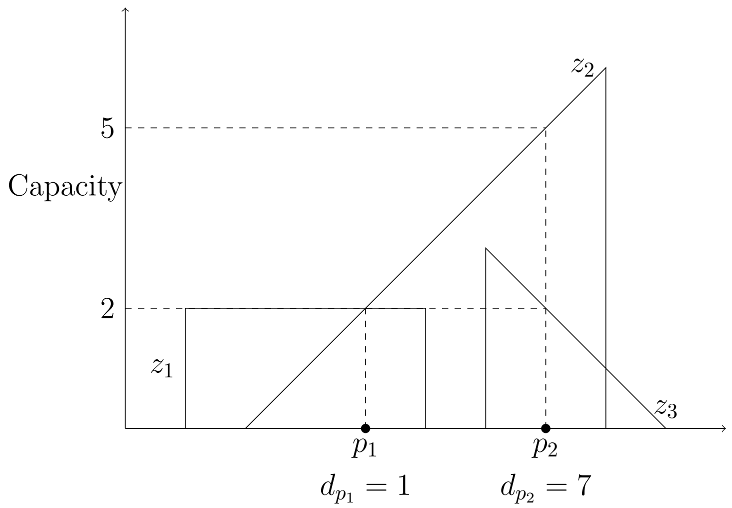
(i) (rectangular) capacity of the form $c_z(p) = c$ for $p \in [a_z, b_z]$, or

(ii) (triangular) capacity of the form $p - a_z$ for $p \in [a_z, b_z]$, or $b_z - p$ for $p \in [a_z, b_z]$.

The goal is to find a subset $Z' \subseteq Z$ of profiles with minimum cost so that the demand of each point $p \in P$ is covered by the capacity in Z', i.e. $\sum_{z\in Z'} c_{z'}(p) \geq d_p$ for all $p \in P$.

Remark: Note that TRC is similar to the UFP-cover problem on a line [4, 12], except that the capacity profiles can also be triangular. Moreover, these triangular profiles are very specific as they rise or fall at slope 1.

[1]This means that a job can be interrupted arbitrarily and resumed on another machine, without any penalty. Simple reductions [15] also show that both preemption and migration are necessary to obtain any non-trivial guarantee.



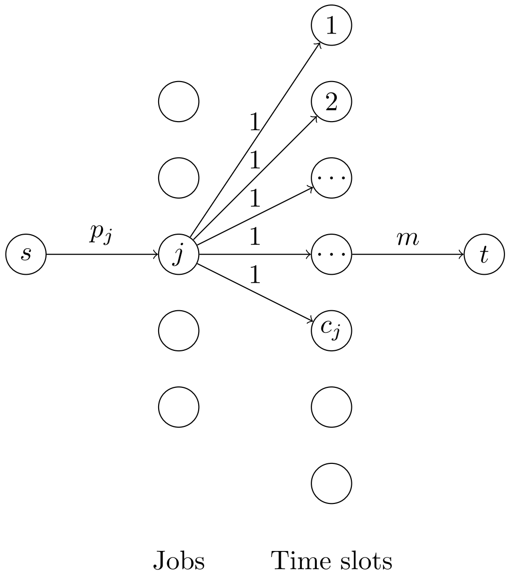


Figure 1: A TRC instance with $Z = \{z_1, z_2, z_3\}$ and two points p_1, p_2. The solution $\{z_2, z_3\}$ is feasible.

Figure 2: The flow network for testing feasibility

We show the following reduction from GSP to TRC.

Theorem 1.3. *An α-approximation for the TRC problem gives a 12α-approximation for GSP.*

The key idea behind this reduction is to view the job-cover inequalities introduced by Moseley [15] in a geometric way. While Moseley uses these inequalities to strengthen the time-indexed LP relaxation, we show that the underlying combinatorial idea can be used directly to relate GSP to the TRC problem. The proof of Theorem 1.3 is described in subsection 3.2. For completeness, we also reprove some related results from [15], that we need here.

1.1.2 Solving the TRC problem

The second main result, which is also the most technical part of the paper is the following.

Theorem 1.4. *There is an LP-based $O(1)$-approximation for the TRC problem.*

While capacitated geometric covering problems are well understood by now [8, 9, 2], a key difference in the TRC problem from those studied previously is that a triangular profile can cover different points by various different amounts. The natural idea is to approximate the triangle by a staircase with $O(\log P)$ different heights[2], and use standard quasi-uniform sampling based approach [16, 9], to get an $O(\log \log P)$ approximation. This is also essentially the main underlying idea behind Moseley's $O(\log \log nP)$ result.

To get around this issue, we partition the instance in a different way and create $O(\log P)$ set multi-cover instances by classifying the point and triangle pairs by the *ratio* of their demands and

[2]Strictly speaking, this requires creating some slack using knapsack-cover inequalities, but we ignore these technical issues for the discussion here

supplies. Solving these multi-cover instances independently would (again) lose $O(\log\log P)$ factor, but we use some geometric properties of these resulting instances, notably that the triangles all have the same slope, to argue that quasi-uniform sampling incurs only an $O(1)$ cumulative loss over all these instances. The precise details are somewhat technical to discuss here, and are deferred to Section 5.

2 Preliminaries

We first formally define the GSP problem we consider. We are given n jobs, indexed $1,\dots,n$, all released at time $t=0$. Each job j has an integer processing time (size) p_j and an arbitrary non-negative, non-decreasing, cost function $f_j:\mathbb{Z}\to\mathbb{Z}$ that specifies the cost of completing the job at time c_j. There are m identical machines, and migration and preemption are allowed. That is, a job can be executed on any (but at most one) machine at any time. The goal is to find a feasible schedule to minimize $\sum_j f_j(c_j)$.

We assume that time is slotted, refer to the interval $(i-1,i]$ as slot i, or time i. As all job sizes are integers, we can assume that at most one job executes during any slot on any machine.

Deadline feasibility and preprocessing. As the objective only depends on the completion times c_j, the GSP problem is equivalent to finding *deadlines* c_j, so that each job can be feasibly scheduled by c_j, and $\sum_j f_j(c_j)$ is minimized. Given candidate deadlines c_j, their feasibility can be easily checked via a flow network [15] (we describe it in subsection 3.1).

We now do some simple preprocessing, so that each job has only $O(\log n)$ candidate deadlines. Losing at most factor 2 in the objective, we can assume that $f_j(x)$ is an integer power of 2 (or 0, or $+\infty$) for all j,x. Let Opt be the value of the optimum solution (we can also assume Opt is finite by first checking that it is feasible to complete all jobs before their deadlines with $+\infty$ cost). Setting $f_j(x)=0$ if $f_j(x)\le \mathrm{Opt}/n^2$, affects the overall solution by a factor of at most $(1-1/n)$. So, by standard doubling and guessing of Opt, and rescaling so that $\mathrm{Opt}=n^2$, it thus suffices to consider only $k=2\log n$ candidate completion times (deadlines) $c_{j,0},\dots,c_{j,k}$ for each job j, where for $i\ge 1$, $c_{j,i}$ is the latest time until which $f_j(x)\le 2^{i-1}$. For $i=0$, $c_{j,0}$ is the latest time until which $f_j(x)=0$.

Knapsack-cover (KC) inequalities. KC inequalities were developed by Carr et al. [7] for the knapsack cover problem. Here, we are given a knapsack with capacity B and items with capacity c_i and weight w_i. The goal is find the minimum weight collection of items that covers the knapsack (has total capacity at least B). The standard LP relaxation

$$\min \sum_i x_i w_i \qquad \text{s.t.} \qquad c_i x_i \ge B \qquad x_i\in[0,1], \quad \forall i\in[n]$$

for this turns out to be arbitrarily bad.

Adding KC inequalities gives the following strengthened formulation,

$$\begin{aligned} &\min \sum_i w_i x_i \\ \sum_{i\notin S} \min(p_i, B-p(S))\, x_i \quad &\ge \quad B-p(S) \qquad \forall S\subset[n],\, p(S)\le B \end{aligned}$$

$$x_i \quad \in \quad [0,1]$$

where $p(S) = \sum_{i \in S} p_i$. Roughly, the inequalities say that even if all the items in S are chosen, the residual demand of $B - p(S)$ still needs to be covered by items not in S.

Carr et al. [7] showed that this reduces the integrality gap to 2, which is also tight, and even though there are exponentially many inequalities, the LP can be solved to any reasonable accuracy in polynomial time. These inequalities have been very useful for various capacitated covering problems [6]. An interesting primal-dual perspective on these inequalites is in [6]. Chakrabarty et al. [8] also give a useful and elegant black-box framework for using these inequalities

3 Geometric Problem

Our goal in this section is to prove Theorem 1.3. To do this, we start with a result of Moseley that considers a flow network for testing the feasibility of deadlines c_j, and gives a succinct description of the min-cuts. Moseley uses this description to derive valid job-cover inequalities for his LP formulation for GSP.

We will instead view these geometrically, and apply a couple of reductions to ultimately reduce GSP to the TRC problem. We start by proving Moseley's result for completeness.

3.1 Deadline Feasibility and Flow Network

Let c_j be the candidate deadline for job j. Consider the following flow network.

The flow network G. See figure 2. There is a source node s and sink node t. Layer 1 consists of n nodes, one for each job j. Layer 2 consists of at most $v = \sum_j p_j$ nodes, one for each possible time slot where a job can execute. There is a directed edge from s to j with capacity p_j. Each job node j has a directed edge of capacity 1 to each time slot in $\{1, \ldots, c_j\}$. From each time slot in $[v]$, there is a directed edge to t with capacity m.

It is clear that a feasible schedule exists if and only if a flow of value $\sum_j p_j$ exists (in fact any feasible integral flow corresponds to a valid schedule, and conversely). In other words, if and only if the minimum s-t cut does not have value less than $\sum_j p_j$.

Note that while the network is of size $O(nP)$ which can be exponentially large, we do not actually solve it algorithmically, and only use it to derive the valid inequalities below.

Moseley showed the following.

Lemma 3.1 ([15]). *Given a set of completion times c_j, there is a feasible schedule if and only if the following condition holds for $b = 0, 1, \ldots, v$.*

$$\sum_{j \in [n]} \min(p_j, \max(c_j - b, 0)) \geq \sum_{j \in [n]} p_j - mb \tag{1}$$

Proof. For a subset $J \subseteq [n]$ of jobs and subset $T \subseteq [v]$ of times slots, let (J, T) denote the s-t cut with the part containing s as $\{s\} \cup J \cup T$, and let $\delta(J, T)$ denote its value. By considering the contribution of each type of edge to $\delta(J,T)$, we have that

$$\delta(J,T) = \sum_{j \notin J} p_j + \sum_{j \in J} |[c_j] \setminus T| + m|T|, \tag{2}$$

where $[c_j] \setminus T$ denotes the time slots in $\{1, \dots, c_j\}$ that do not lie in T.

For any cut (J,T), note that replacing some $t \in T$ with an earlier time $t' < t, t' \notin T$ can only reduce $\delta(J,T)$. Indeed, for $T' = T \cup \{t'\} \setminus \{t\}$, $|T| = |T'|$ and for each j, $|[c_j] \setminus T'| \leq |[c_j] \setminus T|$ as $t' < t$, and hence by (2), $\delta(J,T') \leq \delta(J,T)$. Repeating this process, we can assume that there is some min-cut of the form $\delta(J,[b])$ (possibly with $b = 0$, corresponding to $T = \emptyset$). By (2), we get

$$\delta(J,[b]) = \sum_{j \notin J} p_j + \sum_{j \in J} \max(c_j - b, 0) + mb$$

Finally, for a fixed b, note that each j contributes exactly $\max(c_j - b, 0)$ or p_j or depending on whether $j \in J$ or not. This implies that

$$\min_{J \subseteq [n]} \delta(J,[b]) = \sum_{j \in [n]} \min(p_j, \max(c_j - b, 0)) + mb.$$

This implies that the min-cut is at least $\sum_j p_j$ iff the right side above is at least $\sum_j p_j$ for all $b \geq 0$, giving the claimed result. $\square$

3.2 Reductions to the TRC Problem

The condition (1) for a feasible schedule has a clean geometric view.

Definition 3.2 (Wedge)**.** *For parameters p and c, let a wedge $u_{p,c} : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ be the function $u_{p,c}(t) = \min(p, c - t)$ for $t \leq c$ and 0 otherwise. See figure 3.*

The following simple but useful observation follows directly from the definition above, and Lemma 3.1.

Observation 3.3. *Given completion times c_j, for each job j consider the wedge u_{p_j,c_j}. For $b = 0, 1, \dots,$ let $d_b = \sum_j p_j - mb$ be the demand of point b. Then c_j are feasible iff the wedges satisfy all the demands, i.e. $\sum_j u_{p_j,c_j}(b) \geq d_b$ for all b.*

So GSP reduces (without any loss in objective) to the following *wedge-cover problem.*

Definition 3.4 (Wedge-cover)**.** *For each job j, and every possible deadline c for j, there is a wedge $u_{p_j,c}$ of cost $f_j(c)$. Each point $b = 0, 1, 2, \dots$ has demand $d_b = \sum_j p_j - mb$. Choose* exactly one *wedge for each job j so that the demand of each point is covered.*

3.2.1 Wedge-Cover to Trapezoid-Cover

The reduction above almost leads to a covering problem, except that exactly *one* wedge must be picked for a job (which is a packing condition). But this condition is easily removed to obtain purely covering problem.

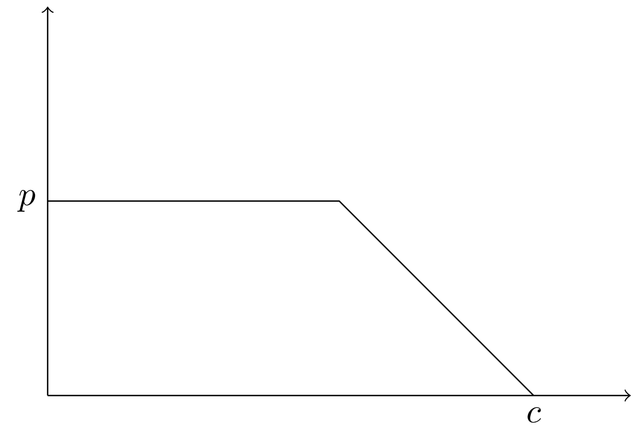


Figure 3: Wedge $u_{p,c}$ for size p and completion time c

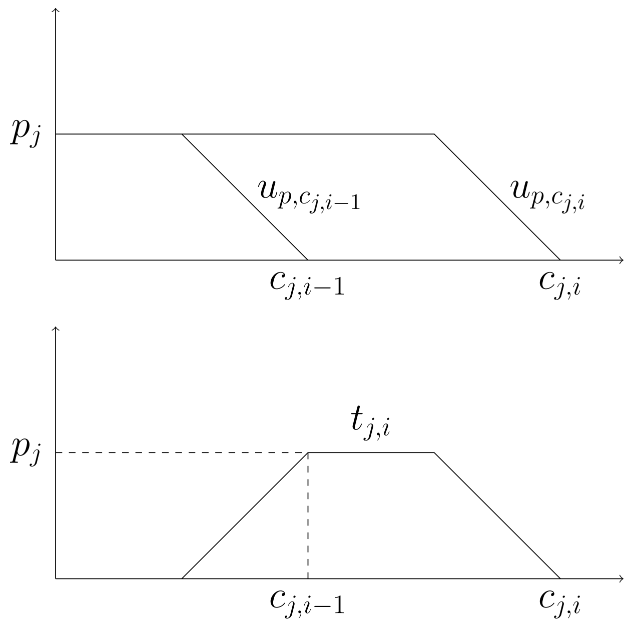


Figure 4: The trapezoid function $t_{j,i}$ obtained by subtracting $u_{p,c_{j,i}}$ and $u_{p,c_{j,i-1}}$

Recall that in GSP problem (after preprocessing) each job j has candidate deadlines $c_{j,i}$ for $i = 1, \dots, k$ with geometrically increasing costs $f_j(c_{j,i}) = 2^{i-1}$. So for each $i = 1, 2, \dots, k$, we replace the wedge $u_{p_j,c_{j,i}}$ by the "trapezoid" function

$$t_{j,i} := u_{p_j,c_{j,i}} - u_{p_j,c_{j,i-1}}$$

(see figure 4) and assign it the cost $f_j(c_{j,i}) = 2^{i-1}$. For $i = 0$, we define $t_{j,0} = u_{p_j,c_{j,0}}$ and assign it cost $f_j(c_{j,0}) = 0$. Note that $t_{j,i}$ has a rectangular part (possibly empty), and possibly two triangular parts, either rising or falling at slope 1. Also, for any point z,

$$u_{p_j,c_{j,i}}(z) = \sum_{g=0}^{i} t_{j,g}(z). \tag{3}$$

Consider the following reduction. Given a GSP instance, we first preprocess it and create $t_{j,i}$ as defined above. Consider the *trapezoid-cover problem* of finding a minimum cost collection of $t_{j,i}$ (possibly several or none, for each job j) so that the demand $d_b = \sum_j p_j - mb$ for each b is satisfied.

Lemma 3.5. *If a GSP instance has a solution of value w, then the corresponding trapezoid-cover instance has a solution of value at most $4w$. Conversely, if the trapezoid-cover instance has a solution of value w, then there is a solution to the GSP instance of value at most w.*

Proof. Consider a GSP instance, and some feasible solution for it with value w. After preprocessing (in particular rounding $f_j(c)$ to powers of 2), this solution has cost at most $2w$. By Observation 3.3, the corresponding wedge-cover instance has a solution of the same value. If this wedge-cover solution picks the wedge $u_{p_j,c_{j,i}}$ for job j, we pick the trapezoids $t_{j,g}$ for $g = 0, \dots, i$ in the trapezoid-cover instance. By (3) this gives a feasible solution, and as the costs of $t_{j,g}$ are increasing powers of 2, this is at most twice the cost of the wedge-cover solution.

Conversely, given a trapezoid-cover instance, for each job j let $i(j)$ denote the highest index such that $t_{j,i(j)}$ is picked. If no such trapezoid is chosen, we set $i(j) = 0$. By (3) choosing the wedge $u_{j,c_{j,i(j)}}$, for each j also gives a feasible solution for the wedge-cover instance of cost at most w, which by Observation 3.3 gives a feasible solution of no higher cost to the GSP instance. □

3.2.2 Trapezoid-cover to TRC

Splitting each trapezoid into (at most three) disjoint pieces, consisting of rectangles or slope 1 right-angled triangles, each with the same cost as the original trapezoid, gives following the TRC problem. Even though the TRC instances arising from GSP have a very specific demand function $d_b = v - mb$, we will not use this and consider the problem for arbitrary demands d_b.

Definition 3.6 (The TRC problem)**.** *We are given a set of integer points P on a line, with point $p \in P$ having an integer demand d_p. There is a set Z of "capacity profiles" (or objects), where a profile (or object) $z \in Z$ has cost w_z with either*

(i) (rectangular) capacity of the form $c_z(p) = c$ for $p \in [a_z, b_z]$, or

(ii) (triangular) capacity of the form $p - a_z$ for $p \in [a_z, b_z]$, or $b_z - p$ for $p \in [a_z, b_z]$.

The goal is to find a subset $Z' \subset Z$ of profiles with minimum cost so that demand of each point $p \in P$ is covered by the capacity in Z', i.e. $\sum_{z \in Z'} c_{z'}(p) \geq d_p$ for all $p \in P$.

Theorem 1.3 directly follows from Lemma 3.5.

4 An $O(1)$ approximation for TRC

In the rest of the paper we focus on giving an $O(1)$ approximation for the TRC problem.

LP Formulation. We use z to index the collection of objects (triangular or rectangular profiles), w_z to denote its cost and $c_z(p)$ to denote the capacity for point p. Then the integer programming formulation for the problem is

$$\min \sum_z w_z x_z \qquad \sum_z c_z(p) x_z \geq d_p \quad \forall p, \qquad x_z \in \{0, 1\}$$

Recall that after the preprocessing in Section 2, the number of points p is $O(n \log n)$. It follows from linearity of the profiles that covering the demand at these points suffices.

We relax x_z to lie in $[0, 1]$, and strengthen the underlying LP by adding KC inequalities for each point p to get the following LP,

$$\min \sum_z w_z x_z$$

$$\begin{aligned}\sum_{z\notin S}\min(c_z(p), d_p - c_S(p))\, x_z &\geq d_p - c_S(p) \qquad \forall p, \forall S\subset Z,\, c_S(p)\le d_p\\ x_z &\in [0,1]\end{aligned}$$

where for a subset S of objects, $c_S(p) = \sum_{z\in S} c_z(p)$.

Pre-processing the LP solution. Let x denote some optimum solution to this LP, and let W be its cost. Let S be the set of objects z with $x_z \geq 1/\beta$, where $\beta = \beta_1\beta_2$, where $\beta_1, \beta_2 = O(1)$ whose values will be specified later. We select the objects in S integrally, which incurs cost at most βW.

Let x' be the solution x restricted to the objects in the residual instance $Z \setminus S$. Let $d'_p = \max(0, d_p - c_S(p))$ be the residual demand of point p. Then $x'_z \leq 1/\beta$ for each z, and by the KC inequalites (applied to this set S), x' satisfies

$$\sum_{z\in Z\setminus S}\min(c_z(p), d'_p)x'_z \geq d'_p.$$

Henceforth, we only focus on rounding this residual instance. So to avoid notational clutter, let us use Z to denote the residual objects, and x to denote the solution $\beta_1 x'$ and d_p denote d'_p. So we have that,

$$\sum_z \min(c_z(p), d_p)x_z \geq \beta_1 d_p$$

and the solution x has cost at most $\beta_1 W$, and $x_z \leq 1/\beta_2$ for all $z\in Z$. In particular, we have now ensured that each point is covered by a β_1 slack.

Decomposing into rectangles and triangles. We have three types of objects: rectangles, triangles with slope 1 and -1. We can use the slack to decompose the problem into three disjoint problems: one for each type of object. We assign each point p into one of three sets $R, T(1)$ or $T(-1)$ depending on whether it is covered to extent at least $\beta_1 d_p/3$ by rectangles, triangles with slope 1, or with slope -1 respectively. So it suffices to show how to round x, while losing an $O(1)$ factor, for each of the problems where

$$\sum_z \min(c_z(p), d_p)x_z \geq \beta_1 d_p/3 \tag{4}$$

and z ranges over a single type of object.

The problem of covering with rectangular objects was considered by [8]. In particular, [8] show that any fractional solution satisfying (4) for $\beta_1 \geq 72$, can be rounded to an integral solution with cost $O(1)$ times the LP cost.

So it suffices to give a rounding procedure for covering with triangles. As the problems for slopes 1 and for slopes -1 are identical, we focus on the problem for triangles of slope 1.

5 Capacitated Triangle Cover

Based on the reductions in the previous section, we have the problem of covering point with demands using triangles $\mathcal{T}$ of slope 1. We are given an LP solution x that satisfies

$$\sum_z \min(c_z(p), d_p)x_z \geq \beta_1 d_p/3$$

with $x_z \leq 1/\beta_2$ and our goal is to round x to obtain a subset of triangles $T \subseteq \mathcal{T}$ that satisfies the demands to at least d_p, and incurs $O(1)$ factor loss in the cost.

Let $\mathcal{I}$ denote the given instance. We first round the quantities $u_z(p)$ and d_p to powers of 2. Let $u_z(p)$ be $\min(c_z(p), d_p)$ rounded down to the nearest integer power of 2. We also round up the demands d_p to the nearest integer power of 2. Note that this ensures $u_z(p) \leq d_p$, and we have that

$$\sum_z u_z(p)x_z \geq \beta_1 d_p/12$$

Partitioning into multi-cover instances. We now partition the instance into several multi-cover instances. To this end, we classify the triangle-point incidences by the ratio of their demands and supplies.

We say that a triangle z is in *class* j for point p (denoted by $\mathtt{cl}(z,p) = j$) if $u_z(p)/d_p = 2^{-j}$. Note that since $u_z(p) \leq d_p$, $j \geq 0$, and since $u_z(p)$ and d_p are powers of 2, j is an integer.

For each class j, we create a multi-cover instance $\mathcal{I}_j$ as follows. A point p can be covered only by triangles z such that $\mathtt{cl}(z,p) = j$, and we set the demand of point p to $n_p(j) = \lfloor \max(\sum_{z:\mathtt{cl}(z,p)=j} x_z - 1, 0) \rfloor$.

By definition of $n_p(j)$, it follows that whenever $n_p(j) > 0$, we have that

$$\sum_{z:\mathtt{cl}(z,p)=j} x_z \geq n_p(j) + 1 \qquad \forall p : n_p(j) > 0 \tag{5}$$

Hence, the solution x is a feasible fractional solution for each multi-cover instance $\mathcal{I}_j$, in fact with slack 1.

Roughly speaking, to find an integral solution for the instance $\mathcal{I}$, it will suffice to find a solution that simultaneously satisfies all the instances $\mathcal{I}_j$ for each class j. In fact, it will suffice to satisfy the demands in $\mathcal{I}_j$ with some slack (which will be crucial for the rounding later). This is where we use the $\beta_1/12 \gg 1$ slack in the LP solution.

Let $n'_p(j) = \max(n_p(j) - 2^{j/2}, 0)$ for each point p for instance $\mathcal{I}_j$. Let us first show that satisfying these weaker demands suffices.

Lemma 5.1. *If $T \subseteq \mathcal{T}$ satisfies a demand of at least $n'_p(j)$ for each point p for each instance $\mathcal{I}_j$, then T is a feasible solution for the instance $\mathcal{I}$.*

Proof. Fix a point p. Then, the demand covered by T satisfies,

$$\sum_{z\in T} u_z(p) = \sum_{j\geq 0} \sum_{z\in T:\mathtt{cl}(z,p)=j} 2^{-j} d_p \geq \sum_{j\geq 0} n'_p(j) 2^{-j} d_p \geq \sum_{j\geq 0} (n_p(j) - 2^{j/2}) 2^{-j} d_p$$

$$\geq \sum_{j \geq 0} \big((\sum_{z:\texttt{cl}(z,p)=j} x_z - 2) 2^{-j} - 2^{-j/2} \big) d_p = \sum_{j \geq 0} \sum_{z:\texttt{cl}(z,p)=j} x_z (2^{-j} d_p) - (\sum_{j \geq 0} ((2 \cdot 2^{-j}) + 2^{-j/2})) d_p$$
$$\geq \sum_z u_z(p) x_z - 8 d_p \geq \beta_1 d_p / 12 - 8 d_p \geq d_p,$$

where the last inequality is by choosing β_1 to be sufficiently large. □

By Lemma 5.1, it suffices to show the following.

Theorem 5.2. *There is a rounding procedure that given x satisfying (5) with $x_z \leq 1/\beta_2$, finds an integral solution T satisfying the following.*

1. *For each point p, the (weaker) demand $n'_p(j)$ in each instance $\mathcal{I}_j$ is satisfied.*
2. *Any triangle z is picked in T with probability $O(x_z)$. So the expected cost of T is $O(1)$ times that of x.*

We now describe the rounding procedure to show Theorem 5.2.

5.1 Rounding Procedure

Our rounding procedure is based on the quasi-uniform sampling technique developed by Varadarajan [16] and refined by Chan et al. [9] for geometric set cover problem, where the underlying sets have low *union complexity.* A more refined version of the method for multi-cover problems was later developed by [3].

For clarity of exposition, we first describe the main idea behind the quasi-uniform sampling approach of [16, 9]. Later we will see how to refine to our setting.

Overview of quasi-uniform sampling. The starting point is the following: Given an LP solution with values x_i for set i, we first assume (without losing much) that x_i is an integer multiple of $2^{-\ell}$ for some large enough ℓ. We make $x_i 2^\ell$ copies of each set i, that are called *replicas* and view each of these replicas as having value $2^{-\ell}$. The solution is then rounded in phases, $k = \ell, \ell-1, \ldots, 1, 0$, where in each phase k, the value of a variable is either doubled or set to 0. This ensures that after phase k, the variables have values that are integer multiples of 2^{-k}, eventually becoming integral for $k = 0$. A set is picked in the final integral solution if any of its replicas is rounded to 1.

To ensure that the rounded solution has value comparable to the intial LP solution, in each phase k, a variable is doubled with probability roughly 1/2 (more precisely, $1/2 + \varepsilon_k$ for some suitably small value ε_k). While the covering constraint for each point still holds in expectation, some points may not be covered (to extent 1) due to the randomness. To cover them some sets are forcibly picked (in the integral solution).

[9] show that ε_k can be set in each phase k (details in Lemma 5.7 below), so that

1. For any set i, the probability that it is forced in phase k, is at most $x_i c^{-k}$, for some $c > 1$. So the probability over all the phases k, that set i is forced is at most $\sum_k x_i c^{-k} = O(x_i)$.

2. The probability that a set i is picked at the end, if some replica survives all the sampling steps is $O(x_i \log \Phi(m))$, where Φ depends on the shallow-cell complexity of the set system (see definition 5.3 below).

Our approach. As our $\mathcal{I}_j$ are multi-cover instances, we will use the variant of quasi-uniform sampling due to [3], which guarantees analogous properties to those above in the multi-cover setting.

We first show that the Φ parameter for the instances $\mathcal{I}_j$ is $O(1)$. This is done below in Lemma 5.4. For a single multi-cover instance $\mathcal{I}_j$ this would give an $O(1)$ approximation. However, as our rounding needs to simultaneously satisfy all the instances $\mathcal{I}_j$, we need more care.

We will apply the above quasi-uniform sampling framework, simultaneously to all the instances $\mathcal{I}_j$, so that a set could be forced due to any of the instances $\mathcal{I}_j$. The key idea is that our way of partitioning into the instances $\mathcal{I}_j$ ensures that in any phase k, for any set i, the probability of forcing set i due to any uncovered points in instance $\mathcal{I}_j$ is $O(x_i c^{-j-k})$. Summing up over all the instances j, this probability remains $O(x_i c^{-k})$, which ensures as above, that the overall probability that i is forced overall the phases, is $O(x_i)$.

We now give the details.

5.1.1 Shallow cell complexity

Let U be a universe of n elements, and F be some family of subsets of U. Chan et al. [9] introduced the notion of *shallow cell complexity* of a set system, which is a very useful measure of the complexity of F for set covering problems.

A collection of sets $F = \{S_1, \ldots, S_m\} \subseteq U$, partitions U into *regions*, where a region is an equivalence class of points that are covered by exactly the same sets S_i in F. The *depth* of a region is the number of sets that cover it.

Definition 5.3 (Shallow cell complexity)**.** *Let F be a family of sets on a universe U. The* shallow cell complexity *of F (denoted by* $\mathrm{Scc}(F)$*) is $f(t,h)$ if for any sub-collection F' of t sets in F, the number of regions of U induced by F' of depth at most h is at most $f(t,h)$.*

Chan et al. [9], building on the work of [16], gave an $O(\log \phi(|F|))$ approximation for weighted set cover, for set systems with $\mathrm{Scc}(F) = t\Phi(t)h^{O(1)}$. This was later extended to weighted multi-cover problems [3], and we will use these ideas in subsection 5.1.2.

The following lemma bounds $\mathrm{Scc}(I_j)$ for our triangle cover instances.

Lemma 5.4. *For any fixed j,* $\mathrm{Scc}(I_j) \leq 2th^2$

To prove Lemma 5.4, we first show a structural property of I_j.

As the capacity profile for z has the form $c_z(p) = p - a_z$, for $p \in [a_z, b_z]$, let us identify the triangle z by the interval $[a_z, b_z]$. We call a_z (resp. b_z) the start (resp. end) point of z, and say that z is alive during $[a_z, b_z]$. Let σ be the ordering of all triangles in $\mathcal{I}$ according to their start point a_z. Here is a simple but useful claim.

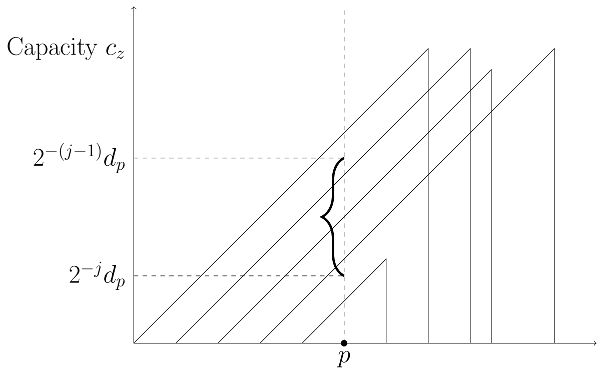


Figure 5: Among all the triangles that cover p, the ones in the instance $\mathcal{I}_j$ form a contiguous sequence when ordered by their starting times.

Claim 5.5. *For any point p, let σ_p be the subsequence of σ restricted to triangles that are alive at p. Then the triangles that cover p in $\mathcal{I}_j$ form a contiguous subsequence*[3] *of σ_p (see Figure 5).*

Proof. A triangle z covers p in the instance $\mathcal{I}_j$, if (i) $p \in [a_z, b_z]$ and (ii) z is of class j for p. i.e. if $u_z(p) = 2^{-j}d_p$, where $u_z(p)$ is equal to $\min\{c_z(p), d_p\}$ rounded down to the nearest integer power of 2.

The first property implies that $z \in \sigma_p$. For $z \in \sigma_p$, the capacity $c_z(p) = (p - a_z)$ is non-increasing in a_z. As taking min with d_p and rounding down to power of 2 preserve the non-increasing property, $u_z(p)$ is also non-increasing, and as σ_p is ordered by a_z, the claim follows. $\square$

By Claim 5.5, to prove Lemma 5.4 it will suffice to bound the following larger quantity by $2th^2$.

Let F' be a sequence of t intervals $[a_z, b_z]$, ordered by increasing a_z (breaking ties arbitrarily). As $|F'| = t$, these intervals divide the line into at most $2t+1$ sub-intervals (equivalence classes of points covered by the same set of intervals), that we index by v. For each such sub-interval v, let S_v be the subsequence of F of intervals that cover v, and let B_v denote the set of contiguous subsequences of S_v of size at most h. Then it suffices to upper bound $|\cup_v B_v|$.

Lemma 5.6. *For B_v are defined above, $|\cup_v B_v| \leq 2th^2$.*

Proof. Let us order the sub-intervals v from left to right and index them $v = 1, 2, \ldots, 2t+1$. So $S_1 = \phi$ and $|B_1| = 0$. It suffices to show that $|B_v \setminus B_{v-1}| \leq h^2$, for all v.

For any v, the sets S_{v-1} and S_v differ by exactly one member z. We consider two cases.

1. If $S_v = S_{v-1} \cup z$. Any contiguous subsequence of S_v that does not contain z is also contiguous in S_{v-1}. On the other hand, the number of contiguous subsequences in S_v containing z and length at most h is at most $\sum_{i=0}^{h-1}(i+1) = h(h+1)/2$, as there are $i+1$ ways of inserting z in a sequence of length i.

2. $S_v = S_{v-1} \setminus z$. Any contiguous subsequence in S_v is either already present in S_{v-1}, or arises by deleting z from some contiguous subsequence in S_{v-1} of length at most $h+1$ (and z not

[3]We avoid using the standard terminology of sub-interval, to avoid confusion with the intervals $[a_z, b_z]$.

at the end). As there are $i-1$ ways of inserting z in a sequence of length i (and not at the end), there are most $\sum_{i=1}^{h}(i-1) = h(h-1)/2$ such subsequences.

□

5.1.2 Quasi-uniform sampling

We now give the details of how to round the solution x_z to the instances $\mathcal{I}_j$.

Instead of working with the fractions x_z, it will be convenient to work with a scaled solution. Let m be the number of variables in the support and ℓ be such that $2m \geq 2^\ell \geq m$.

We assume that there $r_z^{(\ell)} = \lfloor 2^\ell x_z \rfloor$ *replicas* of triangle z. As $x_z \leq 1/\beta_2$, $r_z^{(\ell)} \leq 2^\ell/\beta_2$. As rounding down loses at most one replica per triangle, and there are at most 2^ℓ triangles, the solution still satisfies

$$\sum_{z:\mathtt{cl}(z,p)=j} \min\left(r_z^{(\ell)}, \frac{2^\ell}{\beta_2}\right) \geq 2^\ell n_p(j) \tag{6}$$

Phases. The rounding proceeds in phases $k = \ell, \ell-1, \ldots$ down to some $\alpha = O(1)$. In phase k, each replica is sampled independently with probability $1/2 + \varepsilon_k$. If some point in some instance $\mathcal{I}_j$ is not covered suitably by the replicas after the sampling, some triangles may be *forced* to cover such points. For a forced triangle, all its replicas are removed, and the covering requirement of each point in each instance $\mathcal{I}_j$ is updated accordingly for the next phase.

The final integral solution consists of any triangle that (i) has at least one surviving replica after the last phase α or (ii) was forced in some phase.

Invariant. The algorithm will maintain the following invariant after each phase k.

Let $r_z^{(k)}$ denote the number of surviving replicas of triangle z at the start of phase k (or equivalently, at the end of phase $k+1$). Let $n_p^{(k-1)}(j)$ be the residual demand obtained from $n_p^{(k)}(j)$, by removing the number of forced triangles covering p in instance $\mathcal{I}_j$. For each point p and each instance $\mathcal{I}_j$, if $n_p^{(k-1)}(j) > 2^{j/2}$, then

$$\text{(Inv)} \qquad \sum_{z:\mathtt{cl}(z,p)=j} \min\left(r_z^{(k-1)}, \frac{2^{k-1}}{b_{k-1}}\right) \geq 2^{k-1}\, n_p^{(k-1)}(j). \tag{7}$$

where $b_k \geq 2$ is a slack parameter to be specified later.

Intuitively, this invariant ensures that the residual demand $n_p^{(k-1)}(j)$ is covered by the surviving replicas with scaling 2^{k-1}, even if no triangle can contribute more than $\frac{2^{k-1}}{b_{k-1}}$ replicas.

Forcing Procedure. The key to getting good guarantees using quasi-uniform sampling is to maintain the invariants above while both keeping ϵ_k small, and the forcing probability low. In [3], Bansal and Pruhs gave a forcing procedure, for each phase k, for general multi-cover problems with the following guarantee.

Consider some set system with shallow-cell complexity $Scc(t,h) = t\Phi(t)h^s$. For some integer k, suppose we are given a collection X of replicas with y_z replicas for each set z, satisfying

$$\sum_{z:p\in z} \min\left(y_z, \frac{2^k}{b}\right) \geq 2^k q_p \tag{8}$$

for each point p, where q_p is the demand of p, and $b \geq 2$.

Then there is a forcing procedure that satisfies the following guarantee.

Lemma 5.7. *[3] Let $\varepsilon_k = \gamma\sqrt{\frac{k+\log\Phi(N)}{2^k}}$, for some explicit $\gamma = O_s(1)$ and where $N = |X|$. Let $Y \subseteq X$ be obtained by choosing each replica in X independently with probability $1/2 + \varepsilon_k$. Then there is a forcing procedure that picks some sets Z' and ensures that*

1. *For any set z with $y_z > 0$, $\Pr_Y[z \in Z'] = O(2^{-2k}(\min_p q_p^{-2}))$.*
2. *For each point p,*
$$\sum_{z:p\in z, z\notin Z'} \min\left(\tilde{y}_z, (1+4\varepsilon)\frac{2^{k-1}}{b}\right) \geq 2^{k-1}\tilde{q}_p \tag{9}$$
where $\tilde{y}_z$ is the number of replicas z in Y, and $\tilde{q}_p = q_p - q'_p$, where q'_p is the number of sets in Z' that contain p.

In the light of lemma 5.7, and lemma 5.4, we now define our algorithm for each phase k.

Algorithm for phase k : We set

$$\varepsilon_k = \gamma\sqrt{\frac{k+\log 2}{2^k}} \tag{10}$$

and recursively define b_k as $b_\ell = \beta_2$ and $b_{k-1} = b_k\,(1+4\varepsilon_k)^{-1}$. At the start of phase k, we are given replicas as input, satisfying the invariant for k. We do the following.

1. Sampling. Sample each replica independently with probability $1/2 + \varepsilon_k$.
2. Forcing. For each instance $\mathcal{I}_j$, invoke lemma 5.7 for the set of points p with $n_p^{(k)}(j) > 2^{j/2}$ with parameters $b = b_k$, $q_p = n_p^{(k)}(j)$, $y_z = r_z^{(k)}$ and $\Phi(N) = 2$ to force triangles. Update the residual demands to $n_p^{(k-1)}(j)$. Discard all the replicas of forced triangles.
3. Pruning. If for any triangle z, the number of surviving replicas exceeds $\frac{2^{k-1}}{b_{k-1}}$, keep $\frac{2^{k-1}}{b_{k-1}}$ replicas of z and discard the rest i.e. assign $r_z^{(k-1)} = \min\left(r_z^{(k-1)}, \frac{2^{k-1}}{b_{k-1}}\right)$.

Number of phases: Run the phases $k = \ell, \ell-1, \ell-2, \ldots, \alpha$ as long as $\varepsilon_k \leq \frac{1}{2}$. Note that by (10), $\alpha \approx \log\gamma = O(1)$.

5.2 Analysis

We first show that the invariant holds at the end of every phase. We then show that the final integral solution satisfies the requirements of Theorem 5.2.

We begin with some simple estimates.

Claim 5.8. $\sum_{k=\ell}^{\alpha} \varepsilon_k = O(1)$, *and choosing* β_2 *large enough* $b_k \geq 2$ *for all* $k \geq \alpha$.

Proof. The first bound follows as ε_k is exponentially small in k, and $\varepsilon_\alpha \leq 1/2$. Similarly, $b_k = \beta_2 \prod_{g=\ell}^{k+1} (1+4\varepsilon_g)^{-1} \geq \beta_2 e^{-4\sum_{g=\ell}^{k+1} \varepsilon_g} \geq \beta_2 e^{-\sum_{g=\ell}^{\alpha} \varepsilon_g} \geq 2$ for constant β_2 large enough. □

Lemma 5.9. *The invariant holds at the end of each phase* k *for* $k \geq \alpha$.

Proof. We will apply induction over the phases, and show that if the invariant holds after phase $k+1$ (or equivalently, beginning of phase k), then it also holds after phase k.

In the base case, it holds at the beginning of phase ℓ, as $b_\ell = \beta_2$, and the condition (6) is satisfied.

Suppose that the invariant holds at the beginning of phase k. If for point p and instance $\mathcal{I}_j$, $n_p^{(k)}(j) < 2^{j/2}$, then $n_p^{(k-1)}(j) \leq n_p^{(k)}(j) < 2^{j/2}$ and hence the invariant continues to hold for p, j at the end of phase k.

Else for p, j, $n_p^{(k)}(j) \geq 2^{j/2}$ and the inequality in (7) holds for $k+1$. As $b_k \geq 2$ by Claim 5.8, the condition (8) for Lemma 5.7 holds with $b = b_k$, $y_z = r_z^{(k)}$ and $q_p = n_p^{(k)}(j)$ for the instance $\mathcal{I}_j$.

As we apply the same sampling and forcing procedure as in lemma 5.7 in phase k of our algorithm, it suffices to show that (9) implies the inequality (7) for the invariant after phase k.

To see this, we identify the terms in (9), with those in (7) after phase k. First, $\frac{b}{1+4\varepsilon} = \frac{b_k}{1+4\varepsilon_k}$ which is the same as b_{k-1}. Then, $\min\left(\tilde{y}_z, \frac{2^{k-1}}{b_{k-1}}\right)$ is the number of replicas of z after pruning, and $\tilde{q}_p$ is the residual demand $n_p^{(k-1)}(j)$. This proves (7). □

Lemma 5.10. *The invariant at the end of the algorithm, implies that for each point* p *the weaker demand* $n'_p(j) = \max(n_p(j) - 2^{j/2}, 0)$ *in each instance* $\mathcal{I}_j$ *is satisfied by the integral solution.*

Proof. Consider the solution at the end of phase α. By the definition of the residual demand $n_p^{(\alpha-1)}(j)$, the number of forced triangles covering p in $\mathcal{I}_j$ is $n_p(j) - n_p^{(\alpha-1)}(j)$.

If $n_p^{(\alpha-1)}(j) < 2^{j/2}$, then the demand $n'_p(j)$ is already satisfied by the forced triangles.

On the other hand if $n_p^{(\alpha-1)}(j) \geq 2^{j/2}$ then by our invariant the inequality (7) holds for $k = \alpha$.

As each triangle can contribute at most $\frac{2^{\alpha-1}}{b_{\alpha-1}}$ replicas, and the right side is $2^{\alpha-1} n_p^{(\alpha-1)}(j)$ the number of triangles covering p is at least $n_p^{(\alpha-1)}(j)$. Together with the forced triangles, p is covered by at least $n_p(j) \geq n'_p(j)$ triangles. □

Cost. We now bound the cost of the final integral solution in terms of the LP cost.

The following lemma bounds the forcing probability of any triangle in phase k.

Lemma 5.11. *Fix any phase k, and triangle z such that some replica of z survives after phase $k+1$. Then, the probability that z is forced in phase k is $O(2^{-2k})$.*

Proof. We show that the probability that z is forced due to some point in instance $\mathcal{I}_j$ is $O(2^{-2k-j})$. Summing up over $j \geq 0$ will imply the result.

The forcing procedure for $\mathcal{I}_j$ in phase k only considers points p with $n_p^{(k)}(j) \geq 2^{j/2}$. By lemma 5.7, this implies that the probability that any triangle z is forced is at most $O(2^{-2k}(\min_p : n_p^{(k)}(j)^{-2}))$ which is $O(2^{-2k-j})$. □

We now bound the probability that a replica survives till the end of phase k.

Claim 5.12. *Consider some replica that was present at the beginning of phase ℓ. The probability that this replica survives after phase k is $O\left(2^{-(\ell-k)}\right)$.*

Proof. The probability that the replica survives after phase k is at most

$$\prod_{g=\ell}^{k}(\frac{1}{2}+\varepsilon_g) = 2^{-(\ell-k+1)}\prod_{g=\ell}^{k}(1+2\varepsilon_g) \leq 2^{-(\ell-k+1)}\exp(2\sum_{g=\ell}^{k}\varepsilon_g) = O\left(2^{-(\ell-k)}\right)$$

where the last step in the inequality follows by Claim 5.8. □

We now use Lemma 5.11 and Claim 5.12 to bound the overall cost.

Lemma 5.13. *Any triangle z is chosen in the final integral solution with probability $O(x_z)$.*

Proof. Consider some triangle z. z lies in the final solution if it is either forced in one of the phases, or if one of its replicas survives after phase α. We bound each of these probabilities by $O(x_z)$.

Initially, z has $\lfloor 2^\ell x_z \rfloor$ replicas. By claim 5.12, the probability that some fixed replica of z survives after phase α is $O\left(2^{-(\ell-\alpha)}\right)$. As $\alpha = O(1)$, the probability that any replica of z survives till the end is $2^\ell x_z \cdot O(2^{-(\ell-\alpha)}) = O(x_z)$.

By lemma 5.11, the forcing probability of z in phase k given that some replica of z survived till the end of phase $k+1$ is $O(2^{-2k})$. As z has 2^ℓ x_z replicas initially, and each replica survives till the end of phase $k+1$ with probability $O\left(2^{-(\ell-k-1)}\right)$ by claim 5.12, the total probability that z is forced in phase k is

$$2^\ell x_z \cdot O(2^{-(\ell-k-1)}) \cdot O(2^{-2k}) = O(x_z\ 2^{-k}).$$

Summing over all the phases k, gives an overall forcing probability of $O(x_z)$. □

6 Concluding Remarks

We gave a geometric view of the general scheduling problem on identical machines where jobs have the same release times. We showed how to solve the resulting problem of covering demands on a line by rectangular and triangular capacity profiles, obtaining $O(1)$ approximation for GSP. We leave open the question of handling general release times. While the flow network of Moseley still gives a test for feasibility of deadlines c_j for general release times, it is not clear how to view it as a covering problem, or even how to write an LP with small integrality gap. Another interesting question is to explore the variant of UFP-cover on a line for more general capacity profiles, and more generally, set cover where a set can cover different points by different amounts.

References

[1] Antonios Antoniadis, Ruben Hoeksma, Julie Meißner, José Verschae, and Andreas Wiese. A QPTAS for the general scheduling problem with identical release dates. In *Intl. Colloquium on Automata, Languages, and Programming, ICALP*, pages 31:1–31:14, 2017.

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

[3] Nikhil Bansal and Kirk Pruhs. Weighted geometric set multi-cover via quasi-uniform sampling. *JoCG*, 7(1):221–236, 2016.

[4] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph (Seffi) Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, 2001.

[5] Jatin Batra, Naveen Garg, and Amit Kumar. Constant factor approximation algorithm for weighted flow time on a single machine in pseudo-polynomial time. In *Foundations of Computer Science, FOCS*, pages 778–789, 2018.

[6] Tim Carnes and David B. Shmoys. Primal-dual schema for capacitated covering problems. *Math. Program.*, 153(2):289–308, 2015.

[7] Robert D. Carr, Lisa K. Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Symposium on Discrete Algorithms, SODA*, pages 106–115, 2000.

[8] Deeparnab Chakrabarty, Elyot Grant, and Jochen Könemann. On column-restricted and priority covering integer programs. In *Integer Programming and Combinatorial Optimization, IPCO*, pages 355–368, 2010.

[9] Timothy M. Chan, Elyot Grant, Jochen Könemann, and Malcolm Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *Symposium on Discrete Algorithms, SODA*, pages 1576–1585, 2012.

[10] M. Cheung, J. Mestre, D. Shmoys, and J. Verschae. A primal-dual approximation algorithm for min-sum single-machine scheduling problems. *SIAM Journal on Discrete Mathematics*, 31(2):825–838, 2017.

[11] Uriel Feige, Janardhan Kulkarni, and Shi Li. A polynomial time constant approximation for minimizing total weighted flow-time. In *Symposium on Discrete Algorithms, SODA*, pages 1585–1595, 2019.

[12] Wiebke Höhn, Julián Mestre, and Andreas Wiese. How unsplittable-flow-covering helps scheduling with job-dependent cost functions. *Algorithmica*, 80(4):1191–1213, April 2018.

[13] Sungjin Im and Benjamin Moseley. Fair scheduling via iterative quasi-uniform sampling. In *Symposium on Discrete Algorithms, SODA*, pages 2601–2615, 2017.

[14] Sungjin Im, Benjamin Moseley, and Kirk Pruhs. Online scheduling with general cost functions. *SIAM J. Comput.*, 43(1):126–143, 2014.

[15] Benjamin Moseley. Scheduling to approximate minimization objectives on identical machines. In *International Colloquium on Automata, Languages, and Programming, ICALP*, pages 86:1–86:14, 2019.

[16] Kasturi Varadarajan. Weighted geometric set cover via quasi-uniform sampling. In *Symposium on Theory of Computing, STOC*, pages 641–648, 2010.