The Geometry of Scheduling

Nikhil Bansal * Kirk Pruhs †

Abstract

We consider the following general scheduling problem: The input consists of n jobs, each with an arbitrary release time, size, and monotone function specifying the cost incurred when the job is completed at a particular time. The objective is to find a preemptive schedule of minimum aggregate cost. This problem formulation is general enough to include many natural scheduling objectives, such as total weighted flow time, total weighted tardiness, and sum of flow time squared. We give an $O(\log \log P)$ approximation for this problem, where P is the ratio of the maximum to minimum job size. We also give an $O(1)$ approximation in the special case of identical release times. These results are obtained by reducing the scheduling problem to a geometric capacitated set cover problem in two dimensions.

1 Introduction

We consider the following general offline scheduling problem:

General Scheduling Problem (GSP): The input consists of a collection of n jobs, and for each job j there is a positive integer release time r_j, a positive integer size p_j, and a cost or weight function $w_j(t) \geq 0$ specifying a nonnegative cost for each time $t > r_j$ (we will specify later how these weight functions are represented). A feasible solution is a preemptive schedule, which is an assignment to each unit time interval $[t, t+1]$ of a job j, released not after time t that is run during that time interval. A job is completed once it has been run for p_j units of time. If job j completes at time t, then a cost of $w_j(t)$ is incurred for that job. The objective is to minimize the total cost, $\sum_{j=1}^{n} w_j(c_j)$, where c_j is the completion time of job j.

GSP generalizes several natural scheduling problems, for example:

Weighted Flow Time: If $w_j(t) = w_j \cdot (t - r_j)$, where w_j is some fixed weight associated with job j, then the objective is total weighted flow time.

Flow Time Squared: If $w_j(t) = (t - r_j)^2$, then the objective is the sum of the squares of flow times.

Weighted Tardiness: If $w_j(t) = w_j \max(0, t - d_j)$ for some deadline $d_j \geq r_j$, then the objective is total weighted tardiness.

In general, this problem formulation can model any cost objective function that is the sum of arbitrary cost functions for individual jobs. Note that since preemption is allowed, we can always assume that the cost functions are non-decreasing.

Flow time, which is the duration of time $c_j - r_j$ that a job is in the system, is one of the most natural and commonly used quality of service measures for a job in the computer systems literature. Many commonly-used and commonly-studied scheduling objectives are based on combining the flow times

*Eindhoven Institute of Technology, Eindhoven, the Netherlands. Email: bansal@gmail.com. Supported in part by NWO grant 639.022.211.

†Computer Science Department, University of Pittsburgh, Pittsburgh, PA 15260 USA. Email: kirk@cs.pitt.edu. Supported in part by an IBM Faculty Award, and NSF grants CNS-0325353, IIS-0534531, CCF-0830558, and CCF-1115575.

of the individual jobs. However, flow time is also considered a rather difficult measure to work with mathematically. One reason for this is that even slight perturbations to the input instance, can lead to large changes in the optimum value. Moreover, all the known linear programming relaxations for flow time related measures that we are aware of, have integrality gaps that are polynomially large in n.

Despite much interest, large gaps remain in our understanding for even basic flow time based scheduling objectives. For example, for weighted flow time, the best known approximation ratios achievable by polynomial-time algorithms are poly-logarithmic. For weighted tardiness, and flow time squared, no nontrivial approximation ratios were previously known to be achievable. On the other hand, for all of these three problems, even the possibility of a polynomial time approximation scheme (PTAS) has not been ruled out. We discuss the related previous work further in Section 1.3.

1.1 Our Results

We show the following results.

Theorem 1. *There is a randomized polynomial-time $O(\log \log P)$-approximation algorithm for GSP, where P is the ratio of the maximum to minimum job size.*

Theorem 2. *In the special case when all the jobs have identical release times, there is a deterministic polynomial-time 16-approximation algorithm for GSP.*

Representation of the weight functions: Our algorithms only require that there is an efficient procedure to answer the following type of queries about the weight function: For any job j and integer $q > 0$, what is the earliest time when the cost of completing j is at least q, i.e. what is the smallest t such that $w_j(t) \geq q$.

Clearly, any reasonable representation of the weight function that we are aware of satisfies such a property. In fact, one can weaken this requirement even further. For example, losing a factor 2 in the approximation ratio, we can assume that $w_j(t)$ is always an non-negative integer power of 2, and hence it suffices to be able to answer these queries within an error of $2 - \epsilon$. We also allow $w_j(t)$ to take the value $+\infty$, which can model a hard deadline for j. Assuming such queries are allowed, the running time of our algorithm is polynomial in n and $\log W$. Here W is the maximum value (excluding the value $+\infty$) attained by any weight function.

1.2 Techniques

The key idea behind these results is to view the scheduling problem geometrically, and cast it as a capacitated geometric set cover problem. We then use algorithmic techniques developed for geometric set cover problems, and for capacitated covering problems. In particular, we show that GSP can be reduced (with only a loss of factor 4 in the approximation ratio) to a problem we call R2C, defined below. We then prove Theorem 3 that there is a loglog factor approximation for R2C.

Definition of the R2C Problem: The input contains a collection $\mathcal{P}$ of points in two dimensional space, where each point $p \in \mathcal{P}$ is specified by its coordinates (x_p, y_p) and has an associated positive integer demand d_p. Furthermore, the input contains a collection $\mathcal{R}$ of axis-parallel rectangles, each of them abutting the y-axis, i.e. each rectangle $r \in \mathcal{R}$ has the form $(0, x_r) \times (y_r^1, y_r^2)$. In addition, each rectangle $r \in \mathcal{R}$ has an associated positive integer capacity c_r and positive integer weight w_r. The goal is to find a minimum weight subset of rectangles, such that for each point $p \in \mathcal{P}$, the total capacity of rectangles covering p is at least d_p. Foreshadowing slightly, the rectangle capacities in R2C will correspond to job

sizes in our reduction, thus we also use P to denote the maximum ratio of rectangle capacities. The following is an exact integer programming formulation of the problem.

$$\begin{aligned} & \min \sum w_r x_r && (1) \\ \text{s.t.} \quad & \sum_{r \in \mathcal{R}: p \in r} c_r x_r \geq d_p && \forall p \in \mathcal{P} \\ & x_r \in \{0,1\} && \forall r \in \mathcal{R}. \end{aligned}$$

Theorem 3. *There is a polynomial-time $O(\log \log P)$ approximation algorithm for R2C, where P is the ratio of the maximum to minimum rectangle capacity.*

To prove Theorem 3, we combine recent results on weighted geometric cover problems where the sets have low union complexity, together with approaches for handling capacitated covering problems. Specifically, we crucially use the fact that the rectangles in the R2C problem touch the y-axis, and hence the union complexity of the boundary of any k rectangles (i.e. the number of vertices and edges on the boundary of the union) is $O(k)$.

This low union complexity is very useful. If all the capacities and demands in the R2C instance are 1, i.e. if we consider the standard (uncapacitated) set cover version of R2C, then an $O(1)$ approximation follows from the result of Chan et al. [14], which is a refinement of the breakthrough result of Varadarajan [27] on weighted geometric set cover problems with low union complexity. Thus the $O(\log \log P)$ factor in Theorem 3 actually arises due to the arbitrary capacities and demands in R2C.

To handle arbitrary capacities and demands we use a framework formalized by Chakrabarty et al. [13] based on Knapsack Cover (KC) inequalities [12]. Specifically, [13] show that for any capacitated covering problem, it suffices to design good linear programming (LP) based approximation algorithms for two types of *uncapacitated* problems derived from the original capacitated problem. In particular, given an α upper bound on the integrality gap for the priority cover version of the original problem, and a β upper bound on the integrality gap for the multi-cover version implies an $O(\alpha + \beta)$ integrality gap for the original problem. In the multi-cover version of R2C each point p has an arbitrary integer demand d_p specifying the number of distinct rectangles that must cover it. In the priority cover version of R2C each rectangle and each point has a priority, and each point has to be covered by at least one rectangle of higher priority than itself. Being uncapacitated, these priority and multi-cover problems are often easier to deal with.

We show that the priority version of R2C can be viewed as another geometric (uncapacitated) set cover problem, with the property that the complexity of the union of any k sets is $O(k \log P)$. Since we need this bound on the union complexity of the priority cover problem, we will reprove the results of Chakrabarty et al. [13] and not use them as a black-box. By the result of Chan et al. [14], the $O(k \log P)$ union complexity implies a LP based $\alpha = O(\log \log P)$ approximation for the priority version of $R2C$.

An $O(1)$ approximation for the multi-cover version of R2C follows from the result of Bansal and Pruhs [6], which is an extension of the result of Chan et al. [14] for approximating weighted geometric set cover problems with lower union complexity, to weighted geometric multi-cover problems with low union complexity.

We note that our solution of the R2C problem (and hence the GSP problem) is completely based on LP rounding. Thus one contribution of this work is to provide the first strong linear programming formulation for flow time related problems. This LP is somewhat obscured as we present our results using geometric terminology. However, we will give an explicit description of this scheduling LP in section 4.

Identical Release Times: When all jobs have identical release times, the R2C instance that arises from our reduction from the GSP has a much simpler form. All the points to be covered lie on a line,

and the rectangles are one-dimensional intervals. This is called the generalized caching problem in the literature, and a polynomial-time 4-approximation algorithm is known [7]. Together with our reduction from GSP to R2C, which incurs another factor 4 loss, this implies Theorem 2. Subsequent to this work, Cheung and Shmoys [19] gave a $2+\epsilon$ approximation for GSP with identical release times.

Organization: The paper is organized as follows. In section 2 we give the reduction from GSP to R2C. We also consider here the case of identical release times. In section 3 we give the strengthened LP formulation of R2C based on KC inequalities and for completeness show how rounding this LP solution reduces to rounding the priority cover and multi-cover version of the problem. In sections 3.1 and 3.2 we describe the approximation algorithms for the priority cover and the multi-cover versions of R2C. Finally in section 4, we describe the underlying LP for the scheduling problem explicitly and make some final remarks.

1.3 Related Results

Scheduling: There has been extensive work on various completion time and flow time related objectives in both in the offline and online setting and we refer the reader to [26] for a relatively recent survey. We discuss here some work on special cases of GSP. The most well-studied of these cases is perhaps the weighted flow time. The best known polynomial time algorithms have an approximation ratio of $O(\log^2 P)$ [17], $O(\log W)$ and $O(\log nP)$ [2]. A quasi-PTAS with running time $n^{O_\epsilon(\log P \log W)}$ is also known [16]. In the special case when the weights are the reciprocal of job sizes, the objective is known as average stretch or slow-down, and a PTAS [9, 16] is known.

For weighted tardiness, an $n-1$-approximation algorithm is known for identical release times [18], but nothing seems to be known for arbitrary release dates. A PTAS is possible with the additional restriction that there are only a constant number of deadlines [21], or if jobs have unit size [22]. For total flow time squared, no approximation guarantees are known, unless one uses resource augmentation [5].

Geometric Set Cover: The goal in geometric set cover problems is to improve the general $O(\log n)$ approximation bound for set cover by using the geometric structure. This is an active area of research and various different techniques have been developed. However, until recently most of these techniques applied only to the *unweighted* case. A key idea is the connection between set covers and ϵ-nets [10], where an ϵ-net is a sub-collection of sets that covers all the points that lie in at least an ϵ fraction of the input sets. For typical geometric problems, the existence of ϵ-nets of size at most $(1/\epsilon)g(1/\epsilon)$ implies $O(g(OPT))$-approximate solution for unweighted set cover [10]. Thus, proving better bounds on sizes of ϵ-nets (an active research of research is discrete geometry) directly gives improved guarantees for unweighted set-cover. In another direction, Clarkson and Varadarajan [20] related the guarantee for unweighted set-cover to the union complexity of sets. In particular, if the sets have union complexity $O(kh(k))$, which roughly means that the number of points on the boundary of the union of any collection of k sets is $O(kh(k))$, [20] gives an $O(h(n))$ approximation[1]. This was subsequently improved to $O(\log(h(n))$ [27] and in certain cases extended to the unweighted multi-cover case [15].

However, none of these earlier results apply to weighted case. The problem is that these algorithms are non-uniform in that they sample some sets with much higher probability than that specified by the LP relaxation. In a breakthrough result, Varadarajan [28] designed a new quasi-uniform sampling technique and used it to obtain a $2^{O(\log^* n)}\log(h(n))$ approximation for weighted geometric set cover problems with union complexity $O(kh(k))$. He also gave an improved $O(\log h(n))$ approximation when $h(n)$ grows (possibly quite mildly) with n. Chan et al. [14] refined this algorithm to obtain an $O(\log h(n))$ approximation (for all ranges of $h(n)$) and also stated their result in the more general setting of shallow cell complexity. This result was extended further by Bansal and Pruhs [6] to the multi-cover setting.

[1]The notion of union complexity used by [20] was slightly different that one mentioned here.

Knapsack Cover Inequalities: KC inequalities were developed by Carr et al. [12] for the knapsack cover problem. In this problem, we are given a knapsack with capacity B and items with capacities c_i and weight w_i. The goal is find the minimum weight collection of items that covers the knapsack (i.e., with total capacity at least B). Perhaps surprisingly, the standard LP relaxation for this problem turns out to be arbitrarily bad. Carr et al. [12] strengthened the standard LP for the knapsack cover problem by adding exponentially inequalities and showed that it has an integrality gap of 2. Moreover, this LP can be solved almost exactly in time polynomial in n and $1/\epsilon$ to give an integrality gap of $1+\epsilon$. They also give an explicit polynomial size LP with integrality gap $2+\epsilon$. These inequalities have been very useful for various capacitated covering problems [1, 11, 23, 3, 13]. Recently, Chakrabarty et al. [13] gave an elegant framework for using these inequalities, which allows one to solve capacitated covering problems in blackbox manner without even knowing what the KC inequalities are.

2 The Reduction from GSP to R2C

Our goal in this section is to show the following result.

Theorem 4. *A polynomial-time α-approximation algorithm for R2C implies a polynomial-time 4α approximation algorithm for GSP.*

We now give the reduction from GSP to R2C. Before giving the formal specification of the reduction, we give the background motivation. Since the contribution of a job to the objective only depends on its completion time, instead of specifying a complete schedule, we specify a deadline c_j for each job j by which it must be completed. We call an assignment of deadlines *feasible* if there is a schedule, without loss of generality by scheduling jobs in Earliest Deadline First (EDF) order, where all deadlines are met. In Lemma 6 duality-based condition for feasibility, and then explain how to interpret the condition geometrically.

We need to momentarily digress to discuss our conventions when discussing time. An interval $I=[t_1,t_2]$ consists of time slots $[t_1,t_1+1],\ldots,[t_2-1,t_2]$ and has length $|I|=t_2-t_1$. When we refer to time t, we mean the beginning of slot $[t,t+1]$. Specifically, a job j is completed by time t if it is completed in slot $[t-1,t]$ or earlier, and if a job arrives at time t, then it arrives at beginning of $[t,t+1]$ and can be executed during the slot $[t,t+1]$.

Definition 5. *For an interval $I=[t_1,t_2]$, let $X(I):=\{j : r_j \in I\}$ denote the set of jobs that arrive during I, i.e. $r_j \in \{t_1,\ldots,t_2\}$. We define $\xi(I)$, the excess of I, as $\max(p(X(I))-|I|,0)$, where $p(X(I)):=\sum_{j\in X(I)} p_j$ is the total size of jobs in $X(I)$.*

Lemma 6. *An assignment of deadlines c_j to jobs is feasible if and only if for every interval $I=[t_1,t_2]$, the jobs in $X(I)$ that are assigned a deadline after I have a total size of at least $\xi(I)$. That is,*

$$\sum_{j\in X(I):c(j)>t_2} p_j \geq \xi(I) \qquad \forall I=[t_1,t_2].$$

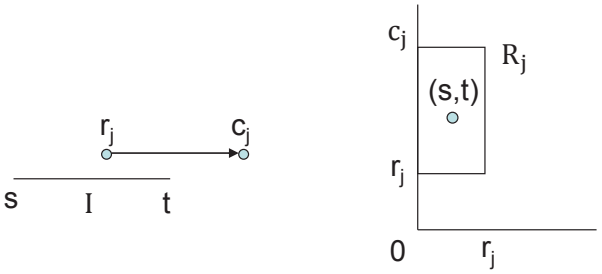
Proof. Consider any interval $I=[t_1,t_2]$. As at most $|I|=t_2-t_1$ amount of work can be done on jobs in $X(I)$ during the interval I, the jobs in $X(I)$ that finish during I can have a total size of at most $|I|$. Thus, the jobs in $X(I)$ that finish after I must have a total size of at least $\max(p(X(I))-|I|,0)=\xi(I)$.

For the converse, we show that if the assignment of deadlines is infeasible, then inequality is violated for some interval I. Consider an infeasible assignment of deadlines and let c_j be the earliest deadline missed when jobs are executed in the EDF order. Let $[t_0-1,t_0]$ be the latest time slot before c_j when EDF was either idle or was working on some job with deadline strictly greater than c_j. Consider the interval $I=[t_0,c_j]$. By definition of t_0, EDF always works on jobs with deadline $\leq c_j$ during I. Moreover, all these jobs arrive during I (as there are no such jobs available at t_0-1), and hence in lie

$X(I)$. As EDF is always working on these jobs during I and still misses the deadline at c_j, it must be that $p(X(I)) > |I|$. □

Geometric View of Lemma 6: Let us associate a point $p_I = (t_1, t_2)$ in two dimensional space with each time interval $I = [t_1, t_2]$. We will view p_I as a witness that enforces that jobs in $X(I)$ finishing after I have total size at least $p(X(I)) - |I|$. So, we associate a demand $d(p_I)$ of $\max(0, p(X(I)) - |I|)$ with I. Next, for each job j and each possible completion time c_j for it, we associate a rectangle $R_j(c_j) = [0, r_j] \times [r_j, c_j - 1]$. We assign $R_j(c_j)$ a cost of $w_j(c_j)$ and capacity of p_j.

We illustrate this view below. On the left is an interval $I = (s, t)$. The job j arrives in I, i.e. $r_j \in I$ and is assigned completion time c_j outside interval I. On the right is the point (s, t) corresponding to the interval I, and the rectangle $R_j = [0, r_j] \times [r_j, c_j - 1]$ corresponding to job j and completion time c_j. Observation 7 notes that (s, t) must be contained in R_j.



Observation 7. *The point p_I lies in the rectangle $R_j(c_j)$ if any only if the job j lies in $X(I)$ and the completion time c_j lies outside (after) I. Moreover, if $p_I \in R_j(c_j)$, then $R_j(c_j)$ contributes exactly p_j towards satisfying the demand of p_I.*

Proof. If a point (s, t) lies in the rectangle $R_j(c_j) = [0, r_j] \times [r_j, c_j - 1]$, this means that $s \in [0, r_j]$ and $t \in [r_j, c_j - 1]$. This is equivalent to the conditions $r_j \in [s, t]$ and $c_j > t$, which is precisely the condition that $j \in X(I)$ where $I = [s, t]$ and has deadline c_j after t. □

By Observation 7 and Lemma 6, GSP can equivalently be stated as: Find the minimum cost collection of rectangles satisfying the following two conditions:

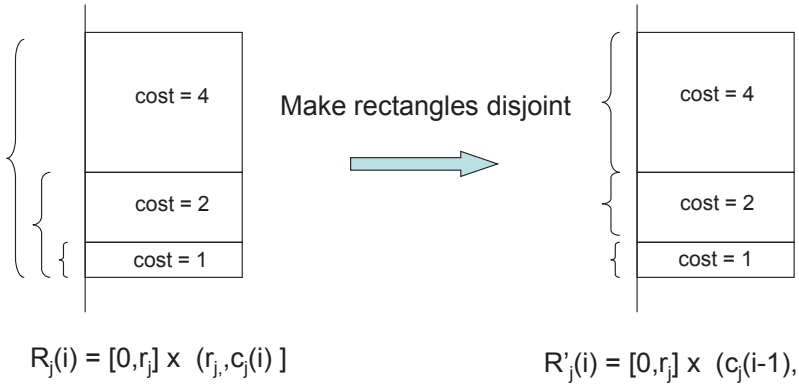
Uniqueness: For each job j, exactly one rectangle of the form $R_j(c_j)$ (for some c_j) is chosen, and

Covering: For each interval I, the demand $d_p(I) = \xi(I)$ of the corresponding point p_I is satisfied.

To get to the R2C problem, we need to show how to remove the uniqueness condition above. Note that in the formulation above, the uniqueness condition is critical, otherwise one may cheat by picking multiple rectangles belonging to the same job (as the demand of a point might be covered by two or more rectangles of the same job, in which case the connection to the scheduling problem breaks down).

To get around this problem, we use another trick (described pictorially in the figure below). By losing a factor of 2 in the approximation ratio, for each job j it suffices to consider only those times c_j when the cost $w_j(c_j)$ first reaches an integer power of 2. Call these times $c_j^0, c_j^1, \ldots$ Now the crucial observation is that $R_j(c') \subset R_j(c)$ for $c' < c$. So, for each possible completion time c_j^i, we define a new modified the rectangle $R'_j(i)$ as $R'_j(i) = R_j(c_j^i) \setminus R_j(c_j^{i-1})$, and give it weight $w(R_j(c))$ and capacity p_j. Now the resulting rectangles $R'_j(i)$ are disjoint, and yet any original rectangle $R_j(c_j^i)$ can be expressed as $R_j(c_j^i) = \cup_{i' \le i} R'_j(i')$. As the costs of $R'_j(i)$ are geometrically increasing, the cost of $\cup_{i' \le i} R'_j(i')$ is at most twice that of $R_j(c_j^i)$. As they are disjoint, using these modified rectangles R'_j instead of R_j will allow us to the uniqueness condition completely.

We now define the reduction formally.



Definition of the Reduction from GSP to R2C: From an arbitrary instance $\mathcal{J}$ of GSP, we create an instance $\mathcal{J}'$ of R2C. Consider $\mathcal{J}$ and some job j. For each integer $k \geq 0$, let I_j^k denote the interval of times (possibly empty) such that $w_j(t) \in [2^k, 2^{k+1})$. Note that for any fixed j, the intervals I_j^k are disjoint and partition the interval $[r_j, \infty)$. Moreover, the number of intervals for any job is $O(\log W)$.

At the loss of factor at most 2 in the objective, we can assume that the deadline c_j for job j is at the right end point of some interval I_j^k. Let $\mathcal{T}$ denote the set of endpoints of intervals I_j^k for all jobs j and indices k. For each job j in $\mathcal{J}$ and $k \geq 0$, create a rectangle $R_j^k = [0, r_j] \times I_j^k$ in $\mathcal{J}'$ with capacity p_j and weight 2^{k+1} . Note that the rectangles $R_j^0, R_j^1, \ldots$ corresponding to a job j are pairwise disjoint. Note also that for simplicity, we have have changed our notation for rectangles from the motivational discussion above.

For each time interval $I = [t_1, t_2]$, where $t_1, t_2 \in \mathcal{T}$, create a point p_I in $\mathcal{J}'$ with demand $d_p = \max(0, p(X(I)) - |I|)$, where $p(X(I)) = \sum_{j: r_j \in I} p_j$. □

We now discuss briefly the complexity of the reduction. Let m denote the number of points in the R2C instance. Clearly, $m = O(|\mathcal{T}|^2)$, as the only relevant times one needs to consider while defining the points and rectangles are the release times of jobs and times in $\mathcal{T}$. As there are $O(\log W)$ intervals for each job j, $|\mathcal{T}| = O(n \log W)$. We now show in Lemma 8 and Lemma 9 that this reduction is approximation preserving (within constant factors). There Lemmas then immediately imply Theorem 4.

Lemma 8. *If there is a feasible solution S to a GSP instance $\mathcal{J}$ with objective value v, then there there is a feasible solution S' to the corresponding R2C instance $\mathcal{J}'$ with objective value at most $4v$.*

Proof. Consider solution S, and for each j, let $k(j)$ be the index of the interval $I_j^{k(j)}$ during which j completes, so the cost incurred by j in S is at least $2^{k(j)}$. Consider the solution S' obtained by choosing for each job j, the rectangles $R_j^0, \ldots, R_j^{k(j)}$. Clearly, the cost contribution of j is $\sum_{i=0}^{k(j)} 2^{i+1} \leq 4 \cdot 2^{k(j)}$, and hence at most 4 times that in S.

It remains to show that S' is feasible, i.e. for every point $p_I \in S'$, the total capacity of rectangles covering p_I is at least $d(p_I) = p(X(I)) - |I|$. Suppose $p = (t_1, t_2)$ corresponds to the time interval $I = [t_1, t_2]$. As S is a feasible schedule, by Lemma 6, the total size of jobs in $X(I)$ that finish after I is at least $p(X(I)) - |I|$. By Observation 7, for each job in $X(I)$ that finishes after I, there is some rectangle in $R_j^0, \ldots, R_j^{k(j)}$ that contributes p_j towards satisfying the demand of p_I. Thus the demand of every point p_I is satisfied. □

Lemma 9. *If there is a feasible solution S' to the R2C instance $\mathcal{J}'$ with value v, then there there is a feasible solution S to GSP instance $\mathcal{J}$ with value at most v.*

Proof. Note that for each job j, at least one rectangle R_j^i must be picked in S'. This is because if we consider the point p_I corresponding to the interval $I = [r_j, r_j]$, it has demand equal to $p(I) - |I| =$

$p(X(r_j)) - 0 = p(X(r_j))$, the total size of jobs arriving on r_j. Since it can only be covered by jobs j in $X(r_j)$, and for such any job at most one rectangle can R_j^i can contribute p_j towards the demand of p_I, one rectangle from each job in $X(r_j)$ must be used.

We construct the solution S as follows. For each job j, let $h(j)$ denote the largest index rectangle $R_j^{h(j)}$ that is chosen in S'. Set the deadline c_j for j as the right end point of $I_j^{h(j)}$. The cost of j in the schedule S is at most $2^{h(j)}$, and hence at most the cost of the rectangle $R_{h(j)}^j$ in J'.

The schedule is feasible for the following reason. If point $p = p_I$ is covered by some rectangle R_j^i corresponding to job j, then by Observation 7, $j \in X(I)$ and the deadline c_j for j is after I. As the demand $d(p_I) = \max(p(X(I)) - |I|, 0)$ of each point p_I is satisfied in S', the total size of jobs in $X(I)$ that are assigned deadline after I is at least $d(p_I)$ and hence by Lemma 6 the schedule is feasible. □

2.1 Identical Release Times

We now consider the special case of identical release times, that is, $r_j = 0$ for all j. Here, the reduction becomes much simpler. In particular, for each rectangle R_j^k, the first dimension $[0, r_j]$ becomes irrelevant and we obtain the following problem: For each job j and $k \geq 0$, there is an interval C_k^j corresponding to completion times with cost in the range $[2^k, 2^{k+1})$. This interval has capacity p_j and weight 2^k. All relevant points p_I corresponds to intervals of the form $[0, t]$ for $t \in \mathcal{T}$ and have demand $D - t$, where D is the total size of all the jobs. For each such interval $I = [0, t)$ (instead of a two dimensional representation), we introduce a point $p_I = t$ with demand $d_I = D - t$. The goal is to find a minimum weight subcollection of intervals C_k^j such that covers the demand.

This problem is a special case of the previously studied Generalized Caching Problem, defined as follows: The input consists of arbitrary demands d_t at various time steps $t = 1, \ldots, n$. In addition there is a collection of time intervals $\mathcal{I}$, where each interval $I \in \mathcal{I}$ has weight w_I, size c_I and span $[s_I, t_I]$ with $s_I, t_I \in \{1, \ldots, n\}$. The goal is to find a minimum weight subset of intervals that covers the demand. That is, find the minimum weight subset of intervals $S \subseteq \mathcal{I}$ such that

$$\sum_{I \in S : t \in [s_I, t_I]} c_I \geq d_t \qquad \forall t \in \{1, \ldots, n\}.$$

A polynomial-time 4-approximation algorithm, based on the local ratio technique, for Generalized Caching is given by by Bar-Noy et al. [7]. This algorithm can equivalently be viewed as a primal dual algorithm applied to a linear program with knapsack cover inequalities [8]. Combining this result with Theorem 4 implies a polynomial time 16 approximation algorithm for GSP in the case of identical release times.

3 Solving the R2C problem

In this section we focus on solving the R2C problem. We consider the natural LP formulation for R2C strengthened by knapsack cover inequalities and then show how to round it suitably. Using standard techniques, we show that the problem of rounding this LP reduces to finding a good rounding for two types of uncapacitated covering instances: a so-called priority cover instance, and several multi-cover instances. While we could direct use the framework of [13] here, we prefer to describe the reduction explicitly both for completeness and also since we will crucially need the fact that there are only $O(\log P)$ distinct priorities in the priority cover version, which is only implicit in [13]. Then in subsection 3.1, we give a rounding algorithm that produces a cover of the resulting priority cover instance with cost $O(\log \log P)$ times the LP cost, and in subsection 3.2 we give an algorithm that produces a cover of the resulting multi-cover instances with cost $O(1)$ times the LP cost. An $O(\log \log P)$ approximation for R2C is then obtained by picking a rectangle r in the final solution if it is included in the covers produced in any of the sub instances.

LP Forumulation: Consider the natural LP relaxation of the integer program for $R2C$ given in line (1), obtained by relaxing $x_r \in \{0,1\}$ to $x_r \in [0,1]$. This LP has an arbitrarily large integrality gap, even when $\mathcal{P}$ consist of just a single point (as R2C on a single point instance is precisely the knapsack cover problem [12]). Thus, we strengthen this LP by adding knapsack cover inequalities introduced in [12]. This gives the the following linear program:

$$\min \sum_{r \in \mathcal{R}} w_r x_r$$

$$\text{s.t.} \quad \sum_{r \in \mathcal{R} \setminus S : p \in r} \min\{c_r, \max(0, d_p - c(S))\}\, x_r \;\geq\; d_p - c(S) \qquad \forall p \in \mathcal{P}, S \subseteq \mathcal{R}$$

$$x_r \;\in\; [0,1] \qquad \forall r \in \mathcal{R}$$

Here $c(S)$ denotes the total capacity of rectangles in S. The constraints are valid for the following reason: For any p, and for any subset of rectangles S, even if all the rectangles in S are chosen, at least a demand of $d_p - c(S)$ must be covered by the remaining rectangles. Moreover, truncating the capacity of a rectangle from c_r to $d_p - c(S)$ (in the constraint for point p) does not affect the feasibility of an integral solution. Even though there are exponentially many constraints per point, for any $\epsilon > 0$ a feasible $(1+\epsilon)$-approximate solution can be found using the Ellipsoid algorithm, see [12] for details. We note that the $(1+\epsilon)$ factor loss is only in the cost, and in particular, all the constraints are satisfied exactly.

Residual Solution: Let x be some $(1+\epsilon)$-approximate feasible solution to the LP above, and let OPT denote the objective value. We simplify x as follows. Let β be a small constant, $\beta = 1/12$ suffices. Let S denote the set of rectangles for which $x_r \geq \beta$. We pick all the rectangles in S, i.e. set $x_r = 1$. Clearly, this cost of this set is at most $1/\beta$ times the LP solution.

For each point p, let $S_p = S \cap \{r : r \in \mathcal{R}, p \in r\}$ denote the set of rectangles in S that contain p. Let us consider the residual instance, with rectangles restricted to $\mathcal{R} \setminus S$ and the demand of point p is $d_p - c(S_p)$. If $d_p - c(S_p) \leq 0$, then p is already covered by S and we discard it. Since the solution x satisfied all the knapsack cover inequalities and hence in particular for S_p, we have that

$$\sum_{r \in \mathcal{R} \setminus S_p : p \in r} \min\{c_r, d_p - c(S_p)\}\, x_r \geq d_p - c(S_p) \qquad \forall p$$

Henceforth, we will only use that x satisfies there particular inequalities.

Scale the solution x restricted to $\mathcal{R} \setminus S$ by $1/\beta$ times. Call this solution x'. Note that since $x_r \leq \beta$, it still holds that $x'_r \in [0,1]$. Clearly, x' satisfies

$$\sum_{r \in \mathcal{R} \setminus S_p : p \in r} \min\{c_r, d_p - c(S_p)\} x'_r \geq \frac{d_p - c(S_p)}{\beta} \qquad \forall p.$$

Let us define the new demand d'_p of p as $d_p - c(S_p)$ rounded up to the nearest integer power of 2. Similarly, defined a new capacity c'_r of each rectangle r to be c_r rounded down to the nearest integer power of 2. The solution x' still satisfies,

$$\sum_{r \in \mathcal{R} \setminus S_p : p \in r} \min\{c'_r, d'_p\} x'_r \geq \frac{d'_p}{4\beta} = 3d'_p \qquad \forall p$$

Decomposition into heavy and light points: We call r a class i rectangle if $c'_r = 2^i c'_{\min}$. Similarly, p is a class i point if $d'_p = 2^i c'_{\min}$ (points could have negative classes). Note that the number of classes for rectangles is at most $O(\log P)$. We call a point p *heavy* if it is covered by rectangles with class at least as high as that of p in the LP solution. That is,

$$\sum_{r \in \mathcal{R}' : c'_r \geq d'_p} \min(c'_r, d'_p) x'_r \geq d'_p.$$

Or equivalently if

$$\sum_{r \in \mathcal{R}' : c'_r \geq d'_p} x'_r \geq 1. \tag{2}$$

Otherwise we say that a point is *light*. A light point satisfies:

$$\sum_{r \in \mathcal{R}' : c'_r < d'_p} c'_r x'_r \geq \left(\frac{1}{4\beta} - 1 \right) d'_p = 2d'_p. \tag{3}$$

We note (for later use) that a point p with $d'_p \leq c'_{\min}$ is always heavy.

3.1 Covering Heavy Points

Covering the heavy points by larger class rectangles reduces to the following problem that we call R3U.

Definition of problem R3U: The input consists of a collection $\mathcal{P}$ consisting of points $p = (p_x, p_y, p_z)$ in three dimensional space, and a collection of cuboids of the form $r = [0, x_r] \times [y^1_r, y^2_r] \times [0, z_r]$ and weight w_r. The goal is to find a minimum weight collection of cuboids that cover all points in $\mathcal{P}$. Note that there are no demands and capacities in R3U.

Lemma 10. *The problem of covering heavy points with rectangles of higher class is a special case of R3U.*

Proof. The reduction takes as input the instance $\mathcal{I}'$ for heavy points and the LP solution x', and creates an instance $\mathcal{A}$ of R3U. For each heavy point $p = (x, y) \in \mathcal{I}'$ with demand d'_p, there is a point (x, y, d'_p) in $\mathcal{A}$. For each rectangle $r = [0, x] \times [y_1, y_2]$ in $\mathcal{I}'$ with capacity c'_r, we define a cuboid $C_r = [0, x] \times [y_1, y_2] \times [0, c'_r]$ of weight w_r. Clearly, a point p in $\mathcal{A}$ can be covered by a cuboid C_r if and only if the corresponding point p in $\mathcal{I}'$ is covered by the corresponding rectangle r and the class of r is not smaller than that of p. □

As the LP solution x' satisfies inequality (2) for heavy points, it gives a feasible LP solution to the R3U instance $\mathcal{A}$. We also note that the cuboids in $\mathcal{A}$ have at most $O(\log P)$ different heights, corresponding to the $O(\log P)$ distinct possible values for c'_r.

Definition 11 (Union Complexity)**.** *Given a collection X of n geometric objects, the union complexity of X is number of edges in the arrangement of the boundary of X. For 3-dimensional objects, this is the total number of vertices, edges and faces on the boundary of X.*

We now bound the union complexity of cuboids in $\mathcal{A}$.

Lemma 12. *For any collection of k rectangles of the type $[0, r] \times [s, t]$, the union complexity is $O(k)$.*

Proof. For each rectangle of the form $[0, r] \times [s, t]$ has a side touching the y-axis. Let us view of union of k such rectangles from as we sweep a vertical line from $x = \infty$ to $x = 0$. Consider the vertical faces on the boundary of the union. For any two rectangles a and b, the pattern $abab$ or $baba$ cannot appear. Thus the vertical faces from a Davenport Schinzel sequence of order 2, which has size at most $2k - 1$ (see for example [24], chapter 7). Since the number of vertices is $O(1)$ times the number of faces, the result follows. □

Lemma 13. *The union complexity of any k cuboids in $\mathcal{R}$ is $O(k \log P)$.*

Proof. This directly follows from Lemma 12 and noting that the number of distinct heights is $O(\log P)$. In particular, since the heights of powers of 2, consider the slice of the arrangement between $z = 2^i$ and $z = 2^{i+1}$. This corresponds to union of rectangles of the form $[0, r] \times [s, t]$. □

We now use the following result of [6], which is an extension of the results of [27] and [14].

Theorem 14 ([6])**.** *Let I be an instance of an (uncapacitated) geometric set multi-cover problem on n points, such that the union complexity of every k sets is at most $kh(k)$ for all k. Then there is a polynomial time $O(\log h(n))$ approximation for the problem, and moreover this approximation guarantee holds with respect to the optimum value of the standard linear programming relaxation for the problem.*

As x' is a feasible fractional solution to $\mathcal{A}$, and $h(k) = O(\log P)$ for $\mathcal{A}$ by Lemma 13, applying Theorem 14 allows us to conclude in Lemma 15 that we can obtain an $O(\log \log P)$ approximation for heavy points.

Lemma 15. *The algorithm finds a cover for heavy points of value at most $O(\log \log P)$ times the cost of x' and hence most $O(\log \log P)$ times the cost of the optimum R2C solution.*

3.2 Covering Light Points

We will relate the problem of covering light points to the following R2M problem.

Definition of the R2M Problem: The input consists of a collection of $\mathcal{P}$ points of the form $p = (x_p, y_p)$, each with an integer demand d_p, and a collection of axis-parallel rectangles r of the form $[0, x_r] \times [y_r^1, y_r^2]$, with cost w_r. The goal is to find a minimum cost subset $S \subset \mathcal{R}$ of rectangles, such that each point $p \in \mathcal{P}$ is covered by at least d_p rectangles in S.

Reducing covering light points to instances of R2M: The reduction takes as input the instance $\mathcal{I}'$ for R2C, restricted to light points, and the LP solution x' satisfying inequality (3). As output, it creates $\log P$ instances of R2M, one instance B_ℓ for each capacity class ℓ. The reduction will ensure that an LP based α approximation for R2M implies a cover for light points in I' with cost $O(\alpha)$ times the cost of the LP solution x'.

The instance B_ℓ: For each $\ell = 0, \ldots, \log P$, the points in B_ℓ are all the light points in $\mathcal{I}'$. The rectangles in B_ℓ are the class ℓ rectangles in $\mathcal{I}'$, i.e. with capacity exactly $2^\ell c'_{\min}$. The cost of each rectangle in B_ℓ is the same as in $\mathcal{I}'$. The demand of a point p in B_ℓ is defined as

$$d_p^\ell = \left\lfloor \sum_{r : p \in r, r \in B_\ell} x'_r \right\rfloor .$$

In the instance B_ℓ, the goal is to cover each point $p \in B_\ell$ by at least d_p^ℓ distinct rectangles.

Observation 16. *The choice of demands d_p^ℓ ensures that the solution x' restricted to rectangles in B_ℓ (i.e. the ones with capacity $2^\ell c'_{\min}$) is a valid solution to B_ℓ.*

Proof. The amount of p covered by x' restricted to rectangles in B_ℓ is $\sum_{r:p\in r, r\in B_\ell} x(r)$ which is at least d_p^ℓ. □

Lemma 17. *Suppose there is an LP based α approximation for R2M. Let S_ℓ be any feasible solution to B_ℓ obtained by applying the LP based algorithm to the solution x' (restricted to B_ℓ). Consider the union S of the rectangles picked in the solutions S_ℓ to the instances $\mathcal{B}_\ell$. Then S satisfies the demand of all the light points in $\mathcal{I}'$, and the cost of S is at most α times the cost of x'.*

Proof. As the cost of S_ℓ is at most α time cost of x' restricted to rectangles in B_ℓ, the cost of S is at most α times that of x'. It remains to show that the solution S is feasible. Consider some point p and suppose it lies in class i in $\mathcal{I}'$, i.e. its demand $d'(p) = 2^i c'_{\min}$. Then the extent to which the demand of p is satisfied by $\bigcup_\ell S_\ell$ is at least

$$\sum_{\ell<i} 2^\ell c'_{\min} d_p^\ell = \sum_{\ell<i} 2^\ell c'_{\min} \lfloor \sum_{r:p\in r, r\in B_\ell} x'_r \rfloor \geq \sum_{\ell<i} 2^\ell c'_{\min} \left((\sum_{r:p\in r, r\in B_\ell} x'_r) - 1 \right)$$

$$\geq \left(\sum_{\ell<i} c'_r \sum_{r:p\in r, r\in B_\ell} x'_r \right) - 2^i c'_{\min} \geq 2d'(p) - d'(p) = d'(p)$$

where last inequality follows from inequality (3) and as $d'(p) = 2^i c'_{\min}$. □

By Theorem 14 and Lemma 12, it follows that there is an LP-based $O(1)$-approximation for R2M. Together with Lemma 17 we gave that

Lemma 18. *The algorithm finds a cover for light points of value at most $O(1)$ times the cost of solution x' and hence at most $O(1)$ times the optimum R2C solution.*

Lemma 15 and Lemma 18 together give Theorem 3.

4 Concluding Remarks

We showed how GSP can be viewed as a geometric covering problem, and used this connection to give an $O(\log \log P)$ approximation for it, and a 16 approximation for the case with identical release times. Given the lack of any inapproximability results for GSP, it seems reasonable to conjecture that an $O(1)$ approximation should exist. Recently, Bansal et al. [4] showed that the geometric approach may be inherently lossy. Using the recent breakthrough constructions of Pach and Tardos [25] on lower bounds on size of ϵ-nets for geometric objects, they show that the LP relaxation for a general instance of the R3U problem (which is a special case of the general R2C problem, when capacities of rectangles are well separated) has an integrality gap of $\Omega(\log \log P)$. Thus for the geometric approach to be further useful some additional property in the reduction from GSP to R2C must be used.

One can bypass the geometric approach altogether and give a direct LP relaxation for GSP based on KC inequalities. This was also our original approach to this problem, but our rounding techniques led us to the geometric approach (which is cleaner to present as several geometric results can be used as a black-box). Recently, Cheung and Shmoys [19] worked the LP for GSP directly to give an elegant $2+\epsilon$ primal-dual approximation for identical release dates. We are not aware of any non-constant integrality gap instance for this LP for general release times, and find it plausible that its integrality gap is actually $O(1)$.

Direct LP Formulation of GSP: For each job j and time t we define a variable $x_{j,t}$ that is intended to be 1 if the job is unfinished at time t, and is 0 otherwise. Note that one only needs to define this variable for *relevant* times t (i.e. release times, or when a weight $f_j(t)$ first reaches a power of 2), but it is convenient to assume that it is defined for each t. Furthermore we assume that $f_j(r_j) = 0$. This not affect anything as each job j has $p_j > 0$ and hence is unfinished at r_j. As previously, an interval $I = [s, t]$ consists of the time slots $[s, s+1], \ldots, [t-1, t]$ and $|I| = t - s$. $X(I)$ denotes the set of jobs that arrive during I (i.e. $r_j \in [s, t]$) and $\xi(I) = \max(0, p(X(I)) - |I|)$.

$$\begin{aligned}
\min \sum_j \sum_{t > r_j} x_{j,t}(f_{j,t} - f_{j,t-1}) & \\
\text{s.t.} \qquad x_{j,t} &\leq x_{j,t-1} \qquad \forall j, t > r_j \\
\sum_{j \in X(I) \setminus S} \min(p_j, \xi(I) - p(S)) x_{j,t} &\geq \xi(I) - p(S) \qquad \forall I = [s,t], \forall S \subset X(I) \text{ with } p(S) \leq \xi(I) \\
x_{j,r_j} &= 1 \qquad \forall j \\
x_{j,t} &\geq 0 \qquad \forall j, t \geq r_j.
\end{aligned}$$

Observe that if a job completes at time t, then in the intended solution $x_{j,t} = 0$ and $x_{j,t'} = 1$ for $t' < t$, and hence the contribution to the objective is $\sum_{r_j < t' < t} (f_{j,t} - f_{j,t-1}) = f_j(t) - f_j(r_j) = f_j(t)$. The first set of constraints says that $x_{j,t}$ are monotonically non-increasing, and the third set of constraints says that all the jobs are unfinished when they arrive. The second set of constraints are KC inequalities applied to the constraint that for each interval $I = [s, t]$ the jobs that finish after t must have a total size of at least $p(X(I))$.

Acknowledgments

We greatly thank Alexander Souza, Cliff Stein, Lap-Kei Lee, Ho-Leung Chan, and Pan Jiangwei for several discussions about this research.

References

[1] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Randomized competitive algorithms for generalized caching. In *ACM Symposium on Theory of Computing*, pages 235–244, 2008.

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

[3] Nikhil Bansal, Anupam Gupta, and Ravishankar Krishnaswamy. A constant factor approximation algorithm for generalized min-sum set cover. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1539–1545, 2010.

[4] Nikhil Bansal, Ravishankar Krishnaswamy, and Barna Saha. On capacitated set cover problems. In *APPROX-RANDOM*, pages 38–49, 2011.

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

[6] Nikhil Bansal and Kirk Pruhs. Weighted geometric set multi-cover via quasi-uniform sampling. In *European Symposium on Algorithms*, pages 145–156, 2012.

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

[8] Reuven Bar-Yehuda and Dror Rawitz. On the equivalence between the primal-dual schema and the local ratio technique. *SIAM J. Discrete Math.*, 19(3):762–797, 2005.

[9] Michael A. Bender, S. Muthukrishnan, and Rajmohan Rajaraman. Approximation algorithms for average stretch scheduling. *Journal of Scheduling*, 7(3):195–222, 2004.

[10] Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.

[11] Tim Carnes and David B. Shmoys. Primal-dual schema for capacitated covering problems. In *Conference on Integer Programming and Combinatorial Optimization*, pages 288–302, 2008.

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

[13] Deeparnab Chakrabarty, Elyot Grant, and Jochen Konemann. On column restricted and priority integer covering programs. In *Conference on Integer Programming and Combinatorial Optimization*, 2010.

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

[15] Chandra Chekuri, Kenneth L. Clarkson, and Sariel Har-Peled. On the set multi-cover problem in geometric settings. In *Symposium on Computational Geometry*, pages 341–350, 2009.

[16] Chandra Chekuri and Sanjeev Khanna. Approximation schemes for preemptive weighted flow time. In *ACM Symposium on Theory of Computing*, pages 297–305, 2002.

[17] Chandra Chekuri, Sanjeev Khanna, and An Zhu. Algorithms for minimizing weighted flow time. In *ACM Symposium on Theory of Computing*, pages 84–93, 2001.

[18] T. C. E. Cheng, C. T. Ng, J. J. Yuan, and Z. H. Liu. Single machine scheduling to minimize total weighted tardiness. *European Journal of Operational Research*, 165(2):423 – 443, 2005.

[19] Maurice Cheung and David B. Shmoys. A primal-dual approximation algorithm for min-sum single-machine scheduling problems. In *APPROX-RANDOM*, pages 135–146, 2011.

[20] Kenneth L. Clarkson and Kasturi R. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007.

[21] George Karakostas, Stavros G. Kolliopoulos, and Jing Wang. An FPTAS for the minimum total weighted tardiness problem with a fixed number of distinct due dates. In *International Computing and Combinatorics Conference*, pages 238–248, 2009.

[22] E .L. Lawler. A fully polynomial approximation scheme for the total tardiness problem. *Operations Research Letters*, 1:207–208, 1982.

[23] Retsef Levi, Andrea Lodi, and Maxim Sviridenko. Approximation algorithms for the capacitated multi-item lot-sizing problem via flow-cover inequalities. *Mathematics of Operations Research*, 33(2), 2008.

[24] Jiri Matousek. *Lectures on Discrete Geometry*. Springer, 2002.

[25] János Pach and Gábor Tardos. Tight lower bounds for the size of epsilon-nets. In *Symposium on Computational Geometry*, pages 458–463, 2011.

[26] Kirk Pruhs, Jiri Sgall, and Eric Torng. Online scheduling. In *Handbook on Scheduling*. CRC Press, 2004.

[27] Kasturi R. Varadarajan. Epsilon nets and union complexity. In *Symposium on Computational Geometry*, pages 11–16, 2009.

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