Scheduling to Approximate Minimization Objectives on Identical Machines

Benjamin Moseley
Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, USA
Relational AI, Berkeley, CA, USA
moseleyb@andrew.cmu.edu

Abstract

This paper considers scheduling on identical machines. The scheduling objective considered in this paper generalizes most scheduling minimization problems. In the problem, there are n jobs and each job j is associated with a monotonically increasing function g_j. The goal is to design a schedule that minimizes $\sum_{j\in[n]} g_j(C_j)$ where C_j is the completion time of job j in the schedule. An $O(1)$-approximation is known for the single machine case. On multiple machines, this paper shows that if the scheduler is required to be either non-migratory or non-preemptive then any algorithm has an unbounded approximation ratio. Using preemption and migration, this paper gives a $O(\log\log nP)$-approximation on multiple machines, the *first* result on multiple machines. These results imply the first non-trivial positive results for several special cases of the problem considered, such as throughput minimization and tardiness.

Natural linear programs known for the problem have a poor integrality gap. The results are obtained by strengthening a natural linear program for the problem with a set of covering inequalities we call *job cover inequalities*. This linear program is rounded to an integral solution by building on quasi-uniform sampling and rounding techniques.

2012 ACM Subject Classification Theory of computation; Theory of computation → Approximation algorithms analysis

Keywords and phrases Scheduling, LP rounding, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.86

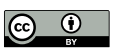
Category Track A: Algorithms, Complexity and Games

Funding *Benjamin Moseley*: Supported in part by a Google Research Award, a Infor Award and NSF Grants CCF-1824303, CCF-1733873 and CCF-1845146.

1 Introduction

A common optimization challenge is scheduling a set of n jobs on m identical machines to optimize the quality of service delivered to the jobs. The quality of service objective could be: a delay based objective, such as minimizing the average waiting time; a fairness objective ensuring resources are shared fairly between jobs, such as the ℓ_2-norm of the waiting time; or a real-time objective such as ensuring a small number of jobs are not completed by their deadline.

Scheduling Model. This paper develops an algorithm that has strong guarantees for most reasonable objectives. This work considers the *identical* machines setting where all jobs are available at the same time. Each job j has a processing time p_j. The job can be processed on m identical machines where the processing time of the job is the same on all machines. This work assumes that *preemption* and *migration* are allowed. That is jobs can be stopped and resumed at a later time, possibly on a different machine.



46th International Colloquium on Automata, Languages, and Programming (ICALP 2019).
Editors: Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi;
Article No. 86; pp. 86:1–86:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

This paper initiates the study of the general scheduling problem (GSP) on identical machines. In this problem, each job j has a function $g_j(t) : \mathbb{R}^+ \to \mathbb{R}^+$. The value of $g_j(t)$ specifies the cost of completing job j at time t. The goal is to design an algorithm that completes each job j at time C_j to minimize $\sum_{j\in[n]} g_j(C_j)$. No assumptions on the functions are made except that they are positive and non-decreasing, so there never is an incentive to have a job wait longer to be completed. Note that each job has its own, *individual*, cost function. In several systems, it is the case that jobs can be associated with distinct cost functions [15, 16, 17].

The problem generalizes many scheduling objectives. Examples include the following. In the following descriptions, each job j has a positive weight w_j denoting its priority.

- **Weighted Completion Time:** A job's cost is its weight multiplied by its completion time. The completion time is how long the job waits in the system and this objective focuses on minimizing the priority scaled average waiting time. This objective is captured by setting $g_j(t) = w_j \cdot t$.
- **Weighted kth Norm of Completion Time:** This objective focuses on minimizing $\sqrt[k]{\sum_{j\in[n]} w_j C_j^k}$ or, by removing the outer kth root, $\sum_{j\in[n]} w_j C_j^k$. This objective is captured by setting $g_j(t) = w_j \cdot t^k$. This is used to enforce fairness in the schedule and typically $k \in \{2, 3\}$.
- **Weighted Throughput Minimization:** The goal is to minimize the weighted number of jobs that miss their deadline. Each job j has a deadline d_j. Setting $g_j(t) = 0$ for $t \leq d_j$ and w_j otherwise gives this objective.
- **Weighted Tardiness:** Each job has no cost if completed before its deadline and otherwise the job pays its weighted waiting time after is deadline. Each job has a deadline d_j and weight w_j. This objective is obtained by setting $g_j(t) = 0$ for $t \leq d_j$ and $w_j(t - d_j)$ otherwise.
- **Exponential Completion Time:** In this objective a job's cost grows exponentially with its completion time. The objective is captured by setting $g_j(t) = w_j \cdot \exp(t)$.

These problems have been challenging to understand. The problem considered is NP-Hard, even on a single machine and the cost functions are piecewise linear [10]. It is known that Smith's rule is optimal for minimizing the total weighted completion time [21]. Bansal and Pruhs in a breakthrough result introduced a $O(1)$-approximation algorithm for the general scheduling problem on a single machine [2]. Cheung et al. improved this to show a $(4 + \epsilon)$ approximation [6, 7, 18]. Antoniadis et al. gave a quasi-polynomial time approximation scheme on a single machine [1, 11].

The next step in this line of work is to generalize these techniques to multiple machine environments, but there is a clear barrier when generalizing past approaches to multiple machines. Prior work introduced a strong linear program that uses a polynomial number of knapsack cover inequalities. See [3] for details on knapsack cover inequalities. The inequalities in [2, 7, 18] are weak in multiple machine environments and result in linear programs with an unbounded integrality gap.

An open question is if there exists a linear program with a small integrality gap for multiple machines. Further, are there good approximation algorithms for the GSP on multiple machines.

Results. This paper studies the GSP in the identical machine environment. The paper shows the following theorem. The technical contributions that result in this theorem are the derivation of valid strong linear program inequalities that are used to strengthen a natural linear program relaxation of the problem and an iterative rounding technique that builds on quasi-uniform sampling [22].

▶ **Theorem 1.** *There is a randomized algorithm that achieves a $O(\log \log nP)$ approximation in expectation and runs in expected polynomial time for the GSP on multiple identical machines with preemption and migration where P is the ratio of the maximum to minimum job size.*

A natural question is if preemption and migration are necessary for an algorithm to have a good approximation ratio. This paper shows that they are by establishing that any scheduler required to be non-preemptive or non-migratory has an unbounded approximation ratio unless $P = NP$. The proof is omitted

▶ **Theorem 2.** *The approximation ratio of any algorithm for GSP is unbounded unless P=NP if either the algorithm is required to be non-migratory or non-preemptive on m identical machines.*

Overview of Technical Contributions. The main result is enabled by a set of strengthening inequalities added to a natural linear program (LP) for the problem. The paper calls these inequalities, **job cover inequalities**. See Section 4. These inequalities are needed because without them the LP introduced in this paper has an unbounded integrality gap even if all jobs arrive at the same time on a single machine[1]. Other natural LP relaxations, such as a time indexed LP, also have an unbounded gap even on a single machine [2, 7].

Prior work on a single machine also used a set of covering inequalities to strengthen a linear program. These inequalities consider every interval I and the set of jobs that arrive during the interval S_I. A constraint states that the total processing time of jobs in S_I that are completed after I ends must be greater than the total processing time of jobs in S_I minus the length of I [2, 7]. This is a covering constraint ensuring that the jobs arriving during I that complete during I have total size at most the length of I. These covering constraints are strengthened using knapsack cover inequalities. If such constraints are satisfied integrally then the Earliest-Deadline-First algorithm can be used to construct a schedule of the same cost as the LP.

A natural idea to extend this to identical machines is to use the same constraint, but the total work completed after I ends must be greater than the size of jobs in S_I minus m times the length of I. This generalization takes into account that each machine can be busy during I. Then the natural next step is to use knapsack cover inequalities to strengthen this new set of constraints. Unfortunately, it is easy to show such inequalities are insufficient and result in an LP with a large integrality gap. There are several issues and they are all rooted in the fact that this does not take into account that a job can only be processed on one machine at any point in time. To overcome this shortcoming, this paper considers covering constraints used to strengthen a minimum cut constraint arises from a natural bipartite flow problem.

This paper proceeds by first reducing the scheduling problem to the problem of finding completion times for each of the jobs, without committing to a schedule. Once a feasible set of completion times is discovered, the scheduling of jobs can easily be obtained by solving a bipartite flow problem. See Section 3 for details. Feasible solutions to the bipartite flow problem have a one-to-one correspondence to the original scheduling problem. We note that the reduction to this flow problem is a well-known scheduling technique.

The bipartite graph in the flow problem is used to derive the job cover inequalities. First an LP is written based on the flow problem. To ensure a feasible flow is possible, a set of constraints is added that ensure the minimum cut in the graph is sufficiently large. Then this

[1] Without strengthening inequalities and when jobs arrive at the same time on a single machine the LP introduced in this paper can be reduced to an LP used in prior work where the gap is known [2, 7]

set of covering constraints are strengthened. While these inequalities are used to strengthen constraints that arise from a flow graph, they are different than previously studied flow cover inequalities [19, 20, 14]. The key to defining the improved constraints is leveraging the structure of the minimum cuts in the bipartite graph resulting from the scheduling problem.

The algorithm solves the strong LP and rounds the solution. The idea is to use iterative randomized rounding, but this results in a large approximation ratio. We remark that standard randomized rounding techniques can be used to obtain a $O(\log n)$ approximation by over sampling variables by a $O(\log n)$ factor and then union bounding over constraints to show they are satisfied with high probability. However, there are issues with reducing the approximation ratio below this factor; the most challenging is showing that the constraints are satisfied if variables are sampled by a smaller factor, which is what would be needed to reduce the approximation ratio.

Instead, the algorithm uses an iterative scheme to round the solution. In each iteration, the algorithm over samples variables by a small factor and modifies the linear program. The modification ensures that (1) in expectation no variable, over all iterations, is sampled by more than a $O(\log \log nP)$ factor than how much it is selected by the optimal LP solution and (2) the relaxation remains feasible. The scheme builds on techniques of quasi-uniform sampling [4, 22] and quasi-uniform iterative rounding [12].

Other Related Work

Single Machine. It is known the Smith's rule is optimal for minimizing the total weighted completion time [21]. The tardiness problem has been challenging to understand. Lawer [13] gave a polynomial approximation scheme if jobs have unit size. Before the work on the GSP in the single machine environment, the previously best known approximation for arbitrary sized jobs was a $(n-1)$-approximation [5]. For the GSP a $(4+\epsilon)$-approximation is known [6] and a quasi-polynomial time approximation scheme has been established [1].

Identical Machines. Smith's rule is optimal for minimizing the total weighted completion time [21]. It is not difficult to see that there is an optimal algorithm for makespan[2] if jobs can be preempted and migrated across machines. An PTAS is known if migration is disallowed [9]. As far as the author is aware, there are no non-trivial results known for exponential completion time and tardiness on multiple machines.

2 Preliminaries

In the general scheduling problem (GSP), there is a set J of n jobs. Each job i has integer processing time p_i. Let P denote the maximum job size and assume the minumum job size is one. The jobs are to be scheduled on a set M of m identical machines that can schedule one job at any point in time. It is assumed that time is slotted and job i must be scheduled for p_i time units over all machines. Jobs can be preempted and migrated across machines. Jobs cannot be scheduled on more than one machine simultaneously. Every job i is associated with a function $g_i : \mathbb{R}^+ \to \mathbb{R}^+$ where $g_i(t)$ specifies the cost of completing job i at time t. Without loss of generality, assume that $g_i(0) = 0$ for all jobs i. The only assumption on $g_i(t)$ is that it is a non-negative non-decreasing function. Under a given schedule, job i is completed at time C_i. The goal is for the scheduler to minimize $\sum_{i \in J} g_i(C_i)$. Note that it can be assumed that all jobs are completed by time nP.

[2] Makespan is equivalent to minimizing the maximum completion.

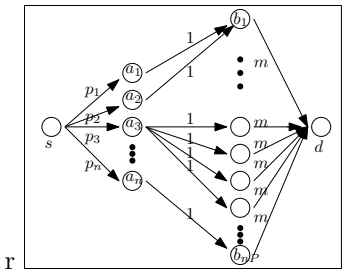


Figure 1 The graph G. A job i needs to assign p_i units of flow (processing time) to machines before C_i. Machines can process up to m units at each time.

3 Scheduling Jobs with Deadlines

This section shows that if the completion times of the jobs are fixed then there is a method to determine how to schedule the jobs at or before their completion times or determine if such a schedule is not possible. Notice that if such a schedule is feasible then this ensures the objective is either the same or smaller in the computed schedule than if all jobs are completed at exactly their given completion times. Let job i have a given completion time C_i. The completion time C_i is interpreted as job i's deadline.

The method to construct a schedule for the jobs is to setup a flow problem. Setting up a flow graph to determine if a set of jobs can be feasibly scheduled is a standard scheduling technique (e.g. [8]), but is presented here so that later this graph can be used in a linear program formulation. Consider creating a bipartite flow graph $G = (\{s, d\} \cup A \cup B, E)$ where A contains a node a_i for every job i and B contains a node b_t for every time step t. There is additionally a source s with an outgoing edge to each node a_i in A with capacity p_i. There is a sink node d that has an incoming edge from each node in B with capacity m. Finally there is an edge from $a_i \in A$ to $b_t \in B$ of capacity 1 if and only if $t \leq C_i$. See Figure 1.

A set of completion times are feasible if and only if there is a feasible flow in this network of value $\sum_{i \in [n]} p_i$. This is because a job can be scheduled for a unit at each time during $[0, C_i]$ and must be scheduled for p_i units total. Further, every time step can schedule up to m jobs.

There are two messages to takeaway from this. One is that the problem can be solved by only knowing completion times for the jobs. The other is that this flow graph can be used to determine if a set of completion times can be associated with a valid schedule. A set of completion times are said to be **valid** if there is a schedule that completes each job only earlier than the given completion time.

The following theorem follows from the construction of G.

▶ **Theorem 3.** *A set of completion times is valid if and only if there is a feasible maximum flow of value $\sum_{i=1}^{n} p_i$ in the flow graph G.*

Given a set of valid completion times, one can construct a feasible schedule using the flow graph. That is, an assignment of jobs to machine at each time step. Unfortunately, the graph has size $\Omega(nP)$ and could be exponential in size. Recall that P is the ratio of the maximum to minimum job size. There is a polynomial time algorithm that constructs a feasible schedule in time polynomial in n and $\log W$ given a set of valid completion times. Here W denotes the maximum value of $g_j(t)$ for $t \leq nP$. This algorithm is omitted due to space. It can be obtained by rounding possible completion times to geometrically increasing times.

4 Strengthened Linear Program with Job Cover Inequalities

Consider the following natural integer program. The constraints in the program come from the flow graph of the prior section. Let $x_{j,t}$ be 1 if job j is not completed at time t and 0 otherwise.[3] This implies that $x_{j,t}$ is continuously 1 for t less than j's completion time and then 0 for times t thereafter in an integer solution. Let T be the set of all time slots and J the set of all jobs. By assuming $g_j(0) = 0$ for all j, the objective function is a telescoping summation for each job j whose value will be $g_j(t)$ if t is the last time j is not fully processed. The first set of constraints says that if job j is completed at time $t-1$ then it is also completed at time t. The second set of constraints are more involved. One can think of the latest time t that $x_{j,t} = 1$ as being the completion time of j. Given this, the constraint says that every cut in the flow network from Section 3 has value at least $\sum_{i \in J} p_i$. This is a valid constraint by Theorem 3 and the maximum-flow minimum-cut theorem. Note that J' corresponds to jobs whose nodes are on the side of the cut with the source s. Similarly, T' corresponds to time steps whose nodes are on the side of the cut with the sink d.

$$\min \sum_{j \in J} \sum_{t \in T} x_{j,t}(g_j(t) - g_j(t-1)) \tag{1}$$

$$\text{s.t.} \quad x_{j,t} \leq x_{j,t-1} \qquad \forall j \in J, t \in T \tag{2}$$

$$\sum_{j \in J \setminus J'} p_j + \sum_{j \in J'} \sum_{t \in T'} x_{j,t} + \sum_{t \in T \setminus T'} m \geq \sum_{j \in J} p_j \qquad \forall J' \subseteq J, T' \subseteq T \tag{3}$$

$$x_{j,t} \in \{0,1\} \qquad \forall j \in J, t \in T$$

The goal is to derive a set of valid strengthening inequalities for the IP (1). These inequalities are used to strengthen the minimum cut constraints. These inequalities are needed because without them the LP has an unbounded integrality gap even if all jobs arrive at the same time on a single machine [7]. Other natural LP relaxations, such as a time indexed LP also have an unbounded gap even on a single machine.

We can derive a set of strengthening inequalities for this linear program that replace the set of constraints (3). The proof establishing validity of the following constraints is omitted due to space. The strengthening focuses on the constraints in (3) for $J' = J$. In the end, the derived constraints are strictly stronger than the above and one need not consider the other sets J'.

Fix $J' = J$ and consider the constraints $\sum_{j \in J} \sum_{t \in T'} x_{j,t} + \sum_{t \in T \setminus T'} m \geq \sum_{j \in J} p_j$ for all $T' \subseteq T$. For a job j and a collection of time steps T' let $E(T', j)$ be the set of up to $\min\{p_j, \sum_{i \in J} p_i - m|T \setminus T'|\}$ **earliest** time steps in T'. The full analysis first shows that we can strengthen this to

$$\sum_{j \in J} \sum_{t \in E(j,T')} x_{j,t} + \sum_{t \in T \setminus T'} m \geq \sum_{j \in J} p_j \text{ for all } T' \subseteq T.$$

Notice the first summation now only considers time steps in $E(T', j)$ for each job j.

The proof further improves these inequalities by taking inspiration from knapsack cover inequalities. Let D be a **vector** where D_j is a time corresponding to job j. Intuitively, D_j is a lower bound on the completion time for job j. The constraints below say that even if all jobs j are set to have completion times **at least** D_j then the constraints should still be satisfied.

[3] Note that this is *not* the standard time indexed LP where the variables represent the amount j is processed at time t.

Fix $T' \subseteq T$ and a vector D of completion times for every job. Consider the constraint for T' in the above set of inequalities. If job j is given a completion time of at least D_j then j will contribute at least $\sum_{t \in [0,D_j] \cap E(T',j)} 1$ to the left side of the inequality. Let $V(T',D) = \sum_{i \in J} p_i - \sum_{t \in T \setminus T'} m - \sum_{j \in J} \sum_{t \in [0,D_j] \cap E(T',j)} 1$. Let $E(T',D,j)$ be the **earliest** $V(T',D)$ time steps in $E(T',j)$ later than D_j. The new constraints are as follows. These are the **job cover inequalities**.

$$\sum_{j \in J} \sum_{t \in E(T',D,j)} x_{j,t} \geq V(T',D) \quad \forall T' \subseteq T, \forall D, V(T',D) > 0 \tag{4}$$

The validity of these inequalities is not difficult to show, but technical. Notice that the number of inequalities is large. We can somewhat reduce the number of constraints as follows. It can be established that any *integer* solution satisfies all the constraints in (4) if and only if the following are satisfied. The proof is omitted.

$$\sum_{j \in J} \sum_{t \in E(T',D,j)} x_{j,t} \geq V(T',D) \quad \forall b \in [0,\infty], T' = [b,\infty], \forall D, V(T',D) > 0 \tag{5}$$

These constraints state that if the subset of constraints in (4) are satisfied for any D and all sets T' that consist of a continuous set of time steps from some time b to time ∞ then all of the constraints in (4) are satisfied (for any T'). Due to this, we will only need to use the constraints in (5) that restricts the sets T'.

The constraints (3) in the IP are replaced by the constraints in (5). Throughout the paper these constraints are discussed and it is said that a fixed constraint is defined by the set $T' = [b,\infty]$ and a vector D.

Note on Solving the LP. The IP is relaxed to a LP. The LP is solved and then subsequently rounded to an integer solution. There are an exponential number of constraints. To solve the LP, the ellipsoid method is used. The author does not know of an efficient separation oracle for the set of constraints in (4). The reduced set of constraints in (5) are the only constraints needed for the analysis. For this set of constraints, an efficient dynamic programming algorithm can be used as a separation oracle. The separation oracle has been omitted due to space constraints.

5 The Rounding Algorithm

In this section the algorithm for rounding a fractional LP solution to an integral solution is described. Recall in Section 3 it was shown how to assign jobs to machines and time slots if a valid set of completion times have been established. The remaining goal is to design an algorithm that rounds a fractional LP solution to an integral solution, giving the completion times for the jobs.

The algorithm takes as input a fractional solution to the LP x'. The input solution is for the LP given in the previous section with constraints (3) replaced with the derived constraints (5). The algorithm rounds the x' solution to an *integral* solution x^*. If this solution is feasible, then the algorithm terminates. Otherwise x^* is modified to obtain a new feasible fractional solution $\tilde{x}^*$ that the algorithm recurses on.

Informal Algorithm Description and Intuition. The algorithm runs in phases. During a phase the algorithm finds a completion time for each job. The completion times are pushed back to later times in each phase. In a fixed phase the algorithm runs a randomized rounding

procedure to sample completion times for the jobs. Roughly, each completion time will be over sampled by a $\Theta(\log \log nP)$ factor over the fractional part of the LP. After sampling an integer solution x^* is created and variables are set corresponding to the sampled completion times of the jobs. Some of the constraints in the LP will be satisfied. These constraints will always remain satisfied because each job's completion time is only pushed back to a later time in subsequent iterations. The algorithm needs to satisfy the remaining constraints, which it does by recursing on a fractional LP solution $\tilde{x}^*$ to make completion times later. The fractional solution $\tilde{x}^*$ is constructed by increasing variables in the integral solution x^*.

In each iteration, the cost of the sampling is bounded by a $\Theta(\log \log nP)$ factor more than the *fractional* portion of the linear program objective in expectation. Ideally, one can use standard iterative rounding for the recursion. However, naive approaches could have large cost as it is difficult to bound the number of times the algorithm recurses and therefore difficult to bound how many times a variable in the LP is sampled. A standard approach will result in a $\Theta(\log nP)$ approximation.

The idea is to only include some fractional variables in the linear program solution that the algorithm recurses on. The variables that are included can be slightly larger than their original value. The essential properties are that (1) the linear program is feasible and (2) each fractional variable is set to 0 with constant probability. Using (2) it will be shown that the expected value of each variable drops by a constant (e.g. $\frac{1}{2}$) factor of its value at the beginning of the iteration. If this is established, then the probability a completion time is sampled decreases geometrically over the phases and we can bound the algorithm's objective by the cost of the sampling in the first phase, a $\Theta(\log \log nP)$ factor within the original LP object in expectation.

The recursion will determine fractional variables to set to satisfy all constraints. One can think of the fractional variables as **fractional completion times** for the jobs. The idea is to associate each constraint with a set of fractional completion times that are *critical* for satisfying the constraint in the solution x'. This will not be all completion times used to satisfy the constraint, but an essential subset of them. By slightly increasing the linear program variables in the integral solution x^* to get a solution $\tilde{x}^*$ it is the case that only the critical completion times are needed to satisfy their corresponding unsatisfied constraints.

If a constraint is unsatisfied, then the solution $\tilde{x}^*$ will set positive fractional values for all completion times critical for this constraint to satisfy it. The analysis will show that each fractional completion time is 0 (not increased) in $\tilde{x}^*$ with constant probability. For a completion time to be 0 in the recursion we need to ensure all constraints are satisfied where the completion time is critical by the integer solution x^*. Unfortunately, a completion time could be critical for many constraints. Due to this, it is insufficient to show each constraint is satisfied with good probability and then union bound over all constraints.

Fix a fractional completion time. The proof establishes that with good probability *every* constraint that the completion time is critical for is satisfied in x^*. This ensures that the the completion is 0 with constant probability. This will be used to show that the expected value of a fractional variable in the LP decreases geometrically over the iterations. A completion time is over-sampled by at most a $\Theta(\log \log nP)$ factor as compared to the original LP solution in expectation over all iterations. The cost is as if only one iteration of uniform sampling occurred. This is similar to the analysis approach used in quasi-uniform sampling [22] and rounding [12].

5.1 Formal Algorithm Description

The ith phase of the algorithm is the following. The algorithm recurses on the following until all constraints are satisfied. Let the input to the first phase be x, the optimal fractional solution to the LP. The algorithm utilizes randomized rounding parameterized by $c \leq 1$. The value of c will be set to be $\frac{1}{\Theta(\log\log nP)}$.

Phase i of the Algorithm. The algorithm first uses **randomized rounding**. Let x' be a feasible fractional solution to the LP at ith phase of the algorithm. For each job j, the algorithm chooses a value $\alpha_j \in [0,1]$ uniformly at random and independently. Let $C_{j,\alpha}$ be the latest time t where $x'_{j,t} \geq c\alpha_j$. Let β'_j be the latest time t where $x'_{j,t} = 1$. Let $\text{LP}'_{\text{int}} = \sum_j g_j(\beta'_j)$ be the total **integral cost** of the LP solution x' and let $\text{LP}'_{\text{frac}} = \sum_{j\in J}\sum_{t>\beta'_j} x'_{j,t}(g_j(t) - g_j(t-1))$ be the total **fractional cost** of the LP solution x'.

The algorithm modifies the LP solution x' to get a new (possibly infeasible) solution x^*. This is further modified to get a feasible solution $\tilde{x}^*$ that the algorithm recurses on. The algorithm sets $x^*_{j,t} = 1$ for all j and t where $t \leq C_{j,\alpha}$ and 0 otherwise. If all constraints are satisfied in the LP, then the algorithm sets $C^*_j = C_{j,\alpha}$ and returns this set of completion times as the final solution. If not, then the algorithm further modifies x^* as described below and recurses on phase $i+1$. Let x^* denote the current integral solution and $\tilde{x}^*$ a fractional solution resulting from the following modification to x^*.

Consider any constraint in (5) defined by a set of time steps T' and a vector of completion times D' that is not satisfied by x^*. Assume D' is chosen so that $D'_j \geq C_{j,\alpha}$ for all j. We only need to consider these constraints because $x_{j,t} = 1$ for $t \leq C_{j,\alpha}$. Recall that we may assume T' contains a continuous set of time steps beginning with some time $t_{T'}$ and going to ∞. The algorithm identifies a set of pairs of jobs and completion times that are fractionally chosen in x' that are "critical" for satisfying this constraint. Intuitively, we will need to include these same fractional completion times in $\tilde{x}^*$.

Let $J_{T',D'}$ be the set of jobs j where $\sum_{t\in E(T',D',j)} x'_{j,t} > 0$. These are jobs used to satisfy the constraint in x'. The following two sets of jobs can be thought of as *not* critical for the constraint.

1. Order the jobs j in $J_{T',D'}$ as $1,2,3,\ldots|J_{T',D'}|$ in decreasing order of D'_j. Let $M_{T',D'} \subseteq J_{T',D'}$ be the smallest prefix of jobs $1,2,\ldots k$ from this order such that $\sum_{j=1}^{k}\sum_{t\in E(T',D',j)} x'_{j,t} \geq \frac{1}{10}V(T',D')$.
2. Order the jobs j in $J_{T',D'}$ as $1,2,3,\ldots,|J_{T',D'}|$ in increasing order of D'_j. Let $L_{T',D'} \subseteq J_{T',D'}$ be the smallest prefix of jobs $1,2,\ldots k$ in this order such that $\sum_{j=1}^{k}\sum_{t\in E(T',D',j)} x'_{j,t} \geq \frac{1}{10}V(T',D')$.

We will say that the jobs in $\mathcal{J}_{T',D'} = J_{T',D'} \setminus (L_{T',D'} \cup M_{T',D'})$ are **critical** for satisfying the constraint T', D'. Let $\mathcal{J}^* = \{j \mid \exists T'D' \text{ s.t. the constraint for } T' \text{ and } D' \text{ is unsatisfied in } x^* \text{ and } j \in \mathcal{J}_{T',D'}\}$. This is the set of all critical jobs for constraints that are not satisfied by x^*.

For each job $j \in \mathcal{J}^*$ set $\tilde{x}^*_{j,t'}$ to be $10x'_{j,t'}$ for all $t' \geq C_{j,\alpha}$. Note that if $c < \frac{1}{10}$ and $C_{j,\alpha} \leq t'$ it is the case that $\tilde{x}^*_{j,t'} \leq c \leq 1/10$, so $\tilde{x}^*$ is in $[0,1]$ as desired. After performing all updates, recurse.

Terminating Condition. The above description states that the algorithm terminates when all constraints are satisfied by the integer solution x^*. The analysis will show that this occurs in polynomial time in expectation.

6 Bounding the Cost and Feasibility of the Algorithm

In this section, the correctness of the algorithm is established and the total cost of the algorithm is bounded as well as the running time. The analysis has several intermediate goals. One is to show that the resulting solution $\tilde{x}^*$ is feasible. Another is to show that the increase in cost of the randomized rounding is bounded by the cost of the fractional part of the LP in one iteration. The final goal is showing that the expected value of the objective for the fractional part of the LP solution decreases significantly in each iteration. After establishing these facts Theorem 3 will prove the main theorem.

Feasibility of the Algorithm. We now establish that $\tilde{x}^*$ is feasible for the LP. This ensures the algorithm constructs a feasible solution. The proof is omitted due to space. The proof follows by the fact that each unsatisfied constraint is associated with variables in the solution x' which are within a constraint multiplicative factor of satisfying the constraint. In the recursion, these variables are included in $\tilde{x}^*$ with their fractional valuesincreased by a factor 10 over x' ensuring their corresponding constraint is satisfied.

▶ **Lemma 4.** *In any iteration of the algorithm, if x' is a feasible LP solution then the algorithm constructs a feasible solution $\tilde{x}^*$ at the end of the iteration for any $c \leq 1/10$.*

Bounding the Cost and Running Time of the Algorithm. This section bounds the cost of the LP solution $\tilde{x}^*$ and the running time of the algorithm. First the cost of the randomized rounding is bounded. This bounds the cost of the *intermediate* integral solution x^*. The following lemma shows that the expected cost *increase* of the LP solution x^* over x' is bounded by $\frac{1}{c}\mathrm{LP}'_{\mathrm{frac}}$. Recall that x^* is the solution obtained by only the randomized rounding part of the algorithm and it is a possibly infeasible integer solution. After this lemma, the cost of the solution $\tilde{x}^*$ is bounded. This is the solution the algorithm recurses on. Combining the cost over the entire algorithm is bounded.

The following lemma bounds the cost of the solution x^*. The proof is omitted. The proof follows by a standard analysis of randomized rounding.

▶ **Lemma 5.** *The total expected different in cost of x^* and x' is at most $\frac{1}{c}\mathrm{LP}'_{\mathrm{frac}}$. That is, $\mathrm{E}[\sum_{j\in J}\sum_t (x^*_{j,t} - x'_{j,t})(g_j(t) - g_j(t-1))] \leq \frac{1}{c}\sum_{j\in J}\sum_t \mathrm{LP}'_{\mathrm{frac}}$.*

Let $\mathrm{LP}^*_{\mathrm{frac}} := \sum_{j\in J}\sum_{t>C_{j,\alpha}} \tilde{x}^*_{j,t}(g_j(t) - g_j(t-1))$ be the fractional cost of $\tilde{x}^*$ and $\mathrm{LP}^*_{\mathrm{int}} = \sum_{j\in J} g(C_{j,\alpha})$ be the integral cost. Note that $\mathrm{LP}^*_{\mathrm{int}}$ is precisely the objective of the integral solution x^*. The key to bounding the cost of the algorithm is to show that the fractional cost of the LP solution decreases by a constant factor in each iteration. This is stated in the following lemma. The proof is deferred due to space. This lemma is the most interesting part of the analysis and the proof is presented in Section 6.1.

▶ **Lemma 6.** *In any iteration of the algorithm, $\mathrm{E}[\mathrm{LP}^*_{\mathrm{frac}}] \leq \frac{1}{4}\mathrm{LP}'_{\mathrm{frac}}$.*

Using this lemma, the total cost of the algorithm can be bounded.

▶ **Lemma 7.** *Let OPT be the optimal feasible objective to the LP. It is the case that the algorithm's total cost is at most $\frac{2}{c}$OPT in expectation.*

Proof. Let $\mathrm{LP}^{\mathrm{i}}_{\mathrm{frac}}$ denote the fractional part of the objective in the LP solution at the beginning of the ith iteration of the algorithm. Note that $\mathrm{LP}^1_{\mathrm{frac}} \leq \mathrm{OPT}$. Inductively, Lemma 6 gives that $\mathrm{E}[\mathrm{LP}^{\mathrm{i}}_{\mathrm{frac}}] \leq \frac{1}{4^i}\mathrm{OPT}$.

Lemma 5 ensures that the integral portion of the LP objective increases by at most $\frac{1}{c}\mathrm{LP}^{\mathrm{i}}_{\mathrm{frac}}$ in each iteration. Thus the total cost can be bounded by $\frac{1}{c}\sum_{i=1}^{\infty}\frac{1}{4^i}\mathrm{OPT} \leq \frac{2}{c}\mathrm{OPT}$. ◀

The following proposition bounds the run time of the algorithm and the proof is omitted. This follows because the previous lemma will ensure the variables in the LP converge to 0 after $O(\log nP)$ iterations.

▶ **Proposition 8.** *The algorithm runs in polynomial time in expectation.*

Lemma 7 bounds the objective of the algorithm. Lemma 4 ensures that the algorithm constructs a feasible solution. Finally, Propostion 8 shows the algorithm runs in polynomial time. Together, this proves the main result, Theorem 1.

6.1 Proof of Lemma 6: Expected Decrease in the Fractional Objective

This section proves Lemma 6 for a fixed iteration of the algorithm. Fix a job j and the fractional solution x' that is input to the rounding algorithm in this iteration. The goal is to show that the probability that j is in $\mathcal{J}^*$ is small and therefore the algorithm only includes integral variables $\tilde{x}^*_{j,t}$ for job j when it recurses. In particular, the goal is to show the following lemma. In the following, n and P are assumed to be sufficiently large.

▶ **Lemma 9.** *Fix any job j^*. With probability at most $\frac{2}{\log^2 nP} \leq 1/40$ it is the case that there is a constraint T', D' unsatisfied by x^* and $j^* \in \mathcal{J}_{T',D'}$ when $c \leq \frac{1}{1000 \log\log nP}$.*

This lemma implies Lemma 6.

Proof of Lemma 6. Fix any job j and an iteration of the algorithm. If there is a constraint T', D' unsatisfied, $j \in \mathcal{J}_{T',D'}$, $x'_{j,t}$ is fractional, $t \geq C_{j,\alpha}$ and $t \in E(T', D', j)$ then the algorithm sets $\tilde{x}^*_{j,t}$ to be $10x'_{j,t}$.

This event increases the cost of $\tilde{x}^*$ by at most $10\sum_{t>C_{j,\alpha}} x'_{j,t}(g_j(t) - g_j(t-1))$ and it happens for some T', D' with probability at most $1/40$. The total expected cost $\mathrm{LP}^*_{\mathrm{frac}}$ is at most the following.

$$10\sum_{j\in J}\sum_{t>C_{j,\alpha}} x'_{j,t}(g_j(t) - g_j(t-1))\Pr[\exists T', D' |\, j \in \mathcal{J}_{T',D'} \text{ and constraint } T', D' \text{ unsatisfied by } x^*]$$

$$\leq 10\sum_{j\in J}\sum_{t>C_{j,\alpha}} x'_{j,t}(g_j(t) - g_j(t-1))\frac{1}{40} \leq \frac{1}{4}\sum_{j\in J}\sum_{t>C_{j,\alpha}} x'_{j,t}(g_j(t) - g_j(t-1)) \quad \text{[Lemma 9]}$$

Thus, the expected cost of $\mathrm{LP}^*_{\mathrm{frac}}$ decreases by a factor $1/4$ over $\mathrm{LP}'_{\mathrm{frac}}$. ◀

The remaining goal of this section is to prove Lemma 9. The proof begins by observing that since the solution x^* is integral if a constraint T' and D' is satisfied for a particular set D' then the constraints for T' and all sets D'' are satisfied. This will allow us to focus on constraints for one special set D'. The proof is omitted.

▶ **Proposition 10.** *Let $t_{T'}$ be some time step and $T' = [t_{T'}, \infty]$. Let D' be set such that D'_j is the latest time t where $x'_{j,t} \geq c$ for all jobs j. If the constraint for T' and D' is satisfied in the solution x^* then all constraints for T' and any set D'' are satisfied in the soultion x^*.*

For the remainder of the proof, fix D' to be as described in the prior lemma. Now we establish some basic propositions on which jobs contribute to a constraint. This will be useful for identifying critical jobs. Note that the two propositions below are different depending on the ordering of the times considered. The proofs are omitted.

▶ **Proposition 11.** *Fix any job j and two sets $T = [t_T, \infty]$ and $T' = [t_{T'}, \infty]$ where $D'_j \leq t_T < t_{T'}$. For any fractional LP solution x satisfying constraints (2) if $V(T, D') \geq \frac{1}{2}V(T', D')$ then it is the case that $\sum_{t \in E(T', D', j)} x_{j,t} \leq 2 \sum_{t \in E(T, D', j)} x_{j,t}$.*

▶ **Proposition 12.** *Fix any job j and two sets $T = [t_T, \infty]$ and $T' = [t_{T'}, \infty]$ and $D'_j > t_T > t_{T'}$. For any fractional LP solution x satisfying constraints (2) if $V(T, D') \geq \frac{1}{2}V(T', D')$ then $\sum_{t \in E(T', D', j)} x_{j,t} \leq 2 \sum_{t \in E(T, D', j)} x_{j,t}$.*

The next lemma establishes that for any fixed constraint T' and D', it is the case that the constraint is satisfied with good probability in the integral solution x^*. Further, the constraint is satisfied if only the jobs in $M_{T', D'}$ (or $L_{T', D'}$) are considered in the summation in the left hand side of the constraint. Due to space, the proof is ommited.

▶ **Lemma 13.** *Fix any $T' = [t_{T'}, \infty]$ and let the vector D' contain the time D'_j that is the latest time t where $x'_{j,t} \geq c$ for all jobs j. With probability at least $1 - \frac{1}{\log^{10} nP}$ it is the case that $\sum_{j \in L_{T', D''}} \sum_{t \in E(T', D', j)} x^*_{j,t} \geq 10V(T', D')$ and $\sum_{j \in M_{T', D'}} \sum_{t \in E(T', D', j)} x^*_{j,t} \geq 10V(T', D')$ when $c \leq \frac{1}{1000 \log \log nP}$.*

Fix any job j^*. Group constraints into two classes for job j^*. The first class are those constraints $T' = [t_{T'}, \infty]$ where $t_{T'} > D'_{j^*}$ and the second class are the remaining constraints. It will be shown separately for both groups of constraints that they are all unsatisfied with small probability.

In the following lemma, the first class of constraints are considered. This lemma heavily relies on Proposition 11.

▶ **Lemma 14.** *Fix any job j^*. With probability at most $1/\log^2 nP$ it is the case that there exists a constraint T', D'' unsatisfied by x^*, $j^* \in \mathcal{J}_{T', D''}$ and $t_{T'} > D'_{j^*}$ when $c \leq \frac{1}{1000 \log \log nP}$.*

Proof. Fix any job j^*. Let D' be set such that D'_j is the latest time t where $x'_{j,t} \geq c$ for all jobs j. The proof will establish that with probability greater $1 - \frac{1}{\log^2 nP}$ it is the case that the constraints for D' and any $T' = [t_{T'}, \infty]$ with $t_{T'} > D'_{j^*}$ are satisfied by x^*. Applying Proposition 10 this implies that x^* satisfies the same allowing for any constraint D'', proving the lemma.

Geometrically group constraints based on the value of $V(T', D')$. Let $\mathcal{C}_k$ contain the set $T' = [t_{T'}, \infty]$ if $j^* \in \mathcal{J}_{T', D'}$, $2^k \leq V(T', D') < 2^{k+1}$ and $t_{T'} > D'_{j^*}$ for any integer $0 \leq k \leq \log nP$.

Fix k and the set $T' \in \mathcal{C}_k$ such that $t_{T'}$ is as late as possible. Let $L_{T', D'}$ be as described in the algorithm definition. We will establish that if $\sum_{j \in L_{T', D'}} \sum_{t \in E(T', D', j)} x^*_{j,t} \geq 10V(T', D')$ then all constraints $V(T'', D')$ for any $T'' \in \mathcal{C}_k$ are satisfied. Once this is established, this will complete the proof as follows. We apply Lemma 13 stating that $\sum_{j \in L_{T', D'}} \sum_{t \in E(T', D', j)} x^*_{j,t} \geq 10V(T', D')$ occurs with probability at least $1 - \frac{1}{\log^{10} nP}$. By union bounding for all $\log nP$ values for k the lemma follows.

Say that $\sum_{j \in L_{T', D'}} \sum_{t \in E(T', D', j)} x^*_{j,t} \geq 10V(T', D')$. Consider any set $T'' \in \mathcal{C}_k$. By definition of the set $L_{T', D'}$ it is the case that $D'_j \leq D'_{j^*}$ for all $j \in L_{T', D'}$. Hence, $D'_j \leq D'_{j^*} \leq t_{T''} \leq t_{T'}$. Thus, Proposition 11 and the geometric grouping of constraints gives that $\sum_{j \in L_{T', D'}} \sum_{t \in E(T'', D', j)} x^*_{j,t} \geq \frac{1}{2} \sum_{j \in L_{T', D'}} \sum_{t \in E(T', D', j)} x^*_{j,t} \geq 5V(T', D') \geq V(T'', D')$. Thus the constraint for T'' and D' is satisfied, proving the lemma. ◀

Similar to the previous lemma, in the following lemma it is shown that all of the second class of constraints are satisfied with good probability. This lemma heavily relies on Proposition 12. This proof is similar to the prior lemma and is omitted.

▶ **Lemma 15.** *Fix any job j^*. With probability at most $1/\log^2 nP$ it is the case that there exists a constraint T', D'' unsatisfied by x^*, $j^* \in \mathcal{J}_{T',D'}$ and $t_{T'} < D'_{j^*}$ when $c \leq \frac{1}{1000 \log\log nP}$.*

For sufficiently large n and P, a union bound and Lemmas 14 and 15 prove Lemma 9.

7 Conclusion

This paper introduced a new set of strong inequalities for scheduling problems on multiple identical machines. Using these inequalities, the paper showed an iterative algorithm that rounds a fractional LP solution to an integral solution which achieves an $O(\log\log nP)$ approximation for most reasonable scheduling minimization problems.

An open question is if there are algorithms with an $O(1)$ approximation ratio for GSP on identical machines. It is also of interest to determine if the inequalities introduced can be extended to other bipartite assignment problems. Can these inequalities be extended to more general environments, such as the related machines or restricted assignment settings? Could a similar analysis be used when jobs arrive over time? These cases introduce new technical hurdles, but $O(1)$ approximation algorithms could be possible by leveraging the given constraints and assuming preemption and migration are allowed.[4] For example, if jobs arrive over time then a generalization of the LP with the strengthened constraints can be derived. However, the rounding becomes challenging because there appears to be no reduction showing only a polynomial number of time sets T' need to be considered in the set of constraints, like was established in this paper. Due to this, it is challenging to show all exponential number of constraints based on sets of times are satisfied by the rounding algorithm.

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 *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, 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. Robert D. Carr, Lisa Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, pages 106–115, 2000.
4. Timothy M. Chan, Elyot Grant, Jochen Könemann, and Malcolm Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1576–1585, 2012.
5. T. C. Edwin Cheng, C. T. Ng, J. J. Yuan, and Zhaohui Liu. Single machine scheduling to minimize total weighted tardiness. *European Journal of Operational Research*, 165(2):423–443, 2005.
6. Maurice Cheung, Julián Mestre, David B. Shmoys, and José Verschae. A Primal-Dual Approximation Algorithm for Min-Sum Single-Machine Scheduling Problems. *SIAM J. Discrete Math.*, 31(2):825–838, 2017.

[4] We remind the reader that any algorithm for GSP on identical machines has an unbounded approximation ratio if either preemption or migration are not allowed

7 Maurice Cheung and David B. Shmoys. A Primal-Dual Approximation Algorithm for Min-Sum Single-Machine Scheduling Problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*, pages 135–146, 2011.

8 Julia Chuzhoy, Sudipto Guha, Sanjeev Khanna, and Joseph Naor. Machine Minimization for Scheduling Jobs with Interval Constraints. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 81–90, 2004.

9 Dorit S. Hochbaum and David B. Shmoys. Using Dual Approximation Algorithms for Scheduling Problems: Theoretical and Practical Results. In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 79–89, 1985.

10 Wiebke Höhn and Tobias Jacobs. On the Performance of Smith's Rule in Single-Machine Scheduling with Nonlinear Cost. *ACM Trans. Algorithms*, 11(4):25, 2015.

11 Wiebke Höhn, Julián Mestre, and Andreas Wiese. How Unsplittable-Flow-Covering Helps Scheduling with Job-Dependent Cost Functions. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 625–636, 2014.

12 Sungjin Im and Benjamin Moseley. Fair Scheduling via Iterative Quasi-Uniform Sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2601–2615, 2017.

13 E.L. Lawler. A fully polynomial approximation scheme for the total tardiness problem. *Operations Research Letters*, 1(6):207–208, 1982. doi:10.1016/0167-6377(82)90022-0.

14 Retsef Levi, Andrea Lodi, and Maxim Sviridenko. Approximation Algorithms for the Capacitated Multi-Item Lot-Sizing Problem via Flow-Cover Inequalities. *Math. Oper. Res.*, 33(2):461–474, 2008.

15 David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. Heracles: Improving Resource Efficiency at Scale. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, pages 450–462, Portland, Oregon, 2015. ACM. doi:10.1145/2749469.2749475.

16 Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, and Mary Lou Soffa. Bubble-Up: Increasing Utilization in Modern Warehouse Scale Computers via Sensible Co-locations. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, pages 248–259, Porto Alegre, Brazil, 2011. ACM. doi:10.1145/2155620.2155650.

17 Paul Marshall, Kate Keahey, and Tim Freeman. Improving Utilization of Infrastructure Clouds. In *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGRID '11, pages 205–214, Washington, DC, USA, 2011. IEEE Computer Society. doi:10.1109/CCGrid.2011.56.

18 Julián Mestre and José Verschae. A 4-approximation for scheduling on a single machine with general cost function. *CoRR*, abs/1403.0298, 2014. arXiv:1403.0298.

19 George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization*. Wiley interscience series in discrete mathematics and optimization. Wiley, 1988.

20 Tony J. Van Roy and Laurence A. Wolsey. Valid inequalities for mixed 0-1 programs. *Discrete Applied Mathematics*, 14(2):199–213, 1986.

21 Wayne E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.

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