The Preemptive Resource Allocation Problem

Kanthi Sarpatwar
IBM T. J. Watson Research Center, Yorktown Heights, NY, United States of America
sarpatwa@us.ibm.com

Baruch Schieber
Computer Science Department, New Jersey Institute of Technology, Newark, NJ, United States of America
https://cs.njit.edu/faculty/sbar
baruch.m.schieber@njit.edu

Hadas Shachnai
Computer Science Department, Technion, Haifa, Israel
hadas@cs.technion.ac.il

Abstract

We revisit a classical scheduling model to incorporate modern trends in data center networks and cloud services. Addressing some key challenges in the allocation of shared resources to user requests (jobs) in such settings, we consider the following variants of the classic *resource allocation problem* (RAP). The input to our problems is a set J of jobs and a set M of homogeneous hosts, each has an available amount of some resource. A job is associated with a release time, a due date, a weight and a given length, as well as its resource requirement. A *feasible* schedule is an allocation of the resource to a subset of the jobs, satisfying the job release times/due dates as well as the resource constraints. A crucial distinction between classic RAP and our problems is that we allow preemption and migration of jobs, motivated by virtualization techniques.

We consider two natural objectives: *throughput maximization* (MaxT), which seeks a maximum weight subset of the jobs that can be feasibly scheduled on the hosts in M, and *resource minimization* (MinR), that is finding the minimum number of (homogeneous) hosts needed to feasibly schedule all jobs. Both problems are known to be NP-hard. We first present an $\Omega(1)$-approximation algorithm for MaxT instances where time-windows form a laminar family of intervals. We then extend the algorithm to handle instances with arbitrary time-windows, assuming there is sufficient slack for each job to be completed. For MinR we study a more general setting with d resources and derive an $O(\log d)$-approximation for any fixed $d \geq 1$, under the assumption that time-windows are not too small. This assumption can be removed leading to a slightly worse ratio of $O(\log d \log^* T)$, where T is the maximum due date of any job.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Packing and covering problems; Theory of computation → Scheduling algorithms

Keywords and phrases Machine Scheduling, Resource Allocation, Vector Packing, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.26

Related Version Full Version: https://arxiv.org/abs/1811.07413

1 Introduction

We revisit a classical scheduling model to incorporate modern trends in data center networks and cloud services. The proliferation of virtualization and containerization technologies, along with the advent of increasingly powerful multi-core processors, has made it possible to execute multiple virtual machines (or *jobs*) simultaneously on the same host, as well as to preempt and migrate jobs with relative ease. We address some fundamental problems in the efficient allocation of shared resources such as CPU cores, RAM, or network bandwidth to several competing jobs. These problems are modeled to exploit multi-job execution and facilitate

preemption and migration, while respecting resource and timing constraints. Typically, the infrastructure service providers are oversubscribed; therefore, the common goals here include admission control of jobs to maximize throughput, or minimizing the additional resource required to process all jobs.

The broad setting considered in this paper is the following. Suppose we are given a set of jobs J that need to be scheduled on a set of identical hosts M, where each host has a limited amount of one or more resources. Each job $j \in J$ has release time r_j, due date d_j, and length p_j, along with a required amount of the resource s_j ($\bar{s}_j$ for multiple resources). A job j can be preempted and migrated across hosts but cannot be processed *simultaneously* on multiple hosts, i.e., at any time a job can be processed by at most one host. However, multiple jobs can be processed by any host at any given time, as long as their combined resource requirement does not exceed the available resource. We consider two commonly occurring objectives, namely, *throughput maximization* and *resource minimization*.

In the *maximum throughput* (MaxT) variant, we are given a set of homogeneous hosts M and a set of jobs J, such that each job j has a profit $w_j > 0$ and attributes (p_j, s_j, r_j, d_j). The goal is to find a subset $S \subseteq J$ of jobs of maximum profit $\sum_{j \in S} w_j$ that can be feasibly scheduled on M. This problem can be viewed as a preemptive variant of the classic *resource allocation problem* (RAP) [26, 11, 8, 5].

In the *resource minimization* (MinR) variant, we assume that each job j has a resource requirement vector $\bar{s}_j \in [0, 1]^d$ as one of the attributes, where $d \geq 1$ is the number of available resources. W.l.o.g., we assume that each host has a unit amount of each of the d resources. A schedule that assigns a set of jobs $S_{i,t}$ to a host $i \in M$ at time t is feasible if $\sum_{j \in S_{i,t}} \bar{s}_j \leq \bar{1}^d$. Given a set of jobs J with attributes $(p_j, \bar{s}_j, r_j, d_j)$, we seek a set of (homogeneous) hosts M of minimum cardinality such that all of the jobs can be scheduled feasibly on M. MinR is a generalization of the classic *vector packing* (VP) problem, in which a set of d-dimensional items needs to be feasibly packed into a minimum number of d-dimensional bins of unit size in each dimension, i.e., the vector sum of all the items packed into each bin has to be less than or equal to $\bar{1}^d$. Any instance of VP can be viewed as an instance of MinR with $r_j = 0$, $d_j = 1$ and $p_j = 1$ for job $j \in J$.

Another application of this general scheduling scenario relates to the allocation of space and time to advertisements by online advertisement platforms (such as Google or Facebook). In the *ad placement problem* [14, 18] we are given a schedule length of T time slots and a collection of ads that need to be scheduled within this time frame. The ads must be placed in a rectangular display area. whose contents can change in different time slots. All ads share the same height, which is the height of the display area, but may have different widths. Several ads may be displayed simultaneously (side by side), as long as their combined width does not exceed the width of the display area. In addition, each advertisement specifies a display count (in the range $1, \ldots, T$), which is the number of time slots during which the ad must be displayed. The actual time slots in which the advertisement will be displayed may be chosen arbitrarily by the scheduler, and, in particular, need not be consecutive. Suppose that each advertisement is associated with some positive profit, and the scheduler may accept or reject any given ad. A common objective is to schedule a maximum-profit subset of ads within a display area of given width. Indeed, this problem can be cast as a special case of MaxT with a single host, where all jobs have the same release time and due date.

1.1 Prior Work

The classical problem of preemptively scheduling a set of jobs with attributes $(p_j, s_j = 1, r_j, d_j)$ on a single machine so as to maximize throughput can be cast as a special case of MaxT with a single host, where each job requires all of the available resource. Lawler [24] showed

that in this special case MaxT admits a *polynomial time approximation scheme (PTAS)*, and the problem is polynomially solvable for uniform job weights. For multiple hosts (i.e., $m = |M| > 1$), this special case of MaxT ($s_j = 1$ for all $j \in J$) admits a $\frac{1}{6+\varepsilon}$-approximation, for any fixed $\varepsilon > 0$. This follows from a result of Kalyanasundaram and Pruhs [21].

As mentioned earlier, another special case of MaxT was studied in the context of advertisement placement. The *ad placement* problem was introduced by Adler *et al.* [1] and later studied in numerous papers (see, e.g., [14, 18, 15, 23, 22] and the survey in [25]). Freund and Naor [18] presented a $(1/3 - \varepsilon)$-approximation for the maximum profit version, namely, for MaxT with a single host and the same release time and due date for all jobs.

Fox and Korupula [17] studied our preemptive scheduling model, with job attributes (p_j, s_j, r_j, d_j), under another popular objective, namely, minimizing weighted flow-time. Their work differs from ours in two ways: while they focus on the online versions, we consider our problems in an offline setting. Further, as they note, while the throughput and resource minimization objectives are also commonly considered metrics, their techniques only deal with flow-time. In fact, these objectives are fundamentally different, and we need novel algorithms to tackle them.

The non-preemptive variant of MaxT, known as the *resource allocation problem* (RAP), was introduced by Phillips *et al.* [26], and later studied by many authors (see, e.g., [7, 6, 8, 9, 19, 11] and the references therein).[1] Chakaravarthy *et al.* [9] consider a generalization of RAP and obtain a constant approximation based on a primal-dual algorithm. We note that the preemptive versus non-preemptive problems differ quite a bit in their structural properties.

As mentioned above, MinR generalizes the classic vector packing (VP) problem. The first non-trivial $O(\log d)$-approximation algorithm for VP was presented by Chekuri and Khanna [10], for any fixed $d \geq 1$. This ratio was improved by Bansal, Caprara and Sviridenko [3] to a randomized algorithm with asymptotic approximation ratio arbitrarily close to $\ln d + 1$. Bansal, Eliás and Khan [4] recently improved this ratio further to $0.807 + \ln(d+1)$. A "fractional variant" of MinR was considered by Jansen and Porkolab [20], where time was assumed to be continuous. For this problem, in the case of a single host, they obtain a PTAS, by solving a configuration linear program (rounding the LP solution is not necessary because time is continuous in their case).

Resource minimization was considered also in the context of the ad placement problem. In this variant, all ads must be scheduled, and the objective is to minimize the width of the display area required to make this possible. Freund and Naor [18] gave a 2-approximation algorithm for the problem, which was later improved by Dawande *et al.* [15] to 3/2. This implies a 3-approximation for MinR instances with $d = 1$, where all jobs have the same release time and due date. We note that this ratio can be slightly improved, using the property that $s_j \leq 1$ for all $j \in J$. Indeed, we can schedule the jobs to use the resource, such that the total resource requirements at any two time slots differ at most by one. Thus, the total amount of resource required at any time exceeds the optimum, OPT, at most by one unit, implying the jobs can be feasibly scheduled on $2OPT + 1$ hosts.

Another line of work relates to the non-preemptive version of MinR, where $d = 1$ and the requirement of each job is equal to 1 (see, e.g. [13, 12]); thus, at most one job can be scheduled on each host at any time.

[1] RAP is also known as the *bandwidth allocation problem*.

1.2 Contributions and Techniques

For summarizing our results, we need the notion of *slackness*. Denote the time window for processing job $j \in J$ by $\chi_j = [r_j, d_j]$, and let $|\chi_j| = d_j - r_j + 1$ denote the length of the interval. Throughout the discussion, we assume that time windows are large enough, i.e., there is a constant $\lambda \in (0, 1)$, such that $p_j \leq \lambda|\chi_j|$ for any job j. Such an assumption is quite reasonable in scenarios arising in our applications. We call λ the *slackness* parameter of the instance.

For the MaxT problem, we present (in Section 3) an $\Omega(1)$-approximation algorithm. As mentioned above, the non-preemptive version of this problem is the classic RAP. To see the structural differences between the non-preemptive and preemptive versions, we consider their natural linear programming relaxations. In solving RAP it suffices to have a variable x_{jt} for each job j and time slot t, indicating the start of job j at slot t. This allows to apply a natural randomized rounding algorithm, where job j is scheduled to start at time t with probability x_{jt}. On the other hand, in MaxT a job can be preempted; therefore, each job requires multiple indicator variables. Further, these variables must be rounded in an *all-or-nothing* fashion, i.e., either we schedule all parts of a job or none of them. Our approach to handle this situation is to, somewhat counter-intuitively, "dumb down" the linear program by not committing the jobs to a particular schedule; instead, we choose a subset of jobs that satisfy certain knapsack constraints and construct the actual schedule in a subsequent phase.

We first consider a *laminar* variant of the problem, where the time windows for the jobs are chosen from a laminar family of intervals.[2] This setting includes several important special cases, such as (i) all jobs are released at $t = 0$ but have different due dates, or (ii) jobs are released at different times, but all must be completed by a given due date. Recall that $m = |M|$ is the number of hosts. Our result for the laminar case is a $\frac{1}{2} - \lambda\left(\frac{1}{2} + \frac{1}{m}\right)$-approximation algorithm, assuming that the slackness parameter satisfies $\lambda < 1 - \frac{2}{m+2}$. Using a simple transformation of an arbitrary instance to laminar, we obtain a $\frac{1}{8} - \lambda\left(\frac{1}{2} + \frac{1}{m}\right)$-approximation algorithm for general instances, assuming that $\lambda < \frac{1}{4} - \frac{1}{2(m+2)}$. Our results imply that as λ decreases, the approximation ratio approaches $\frac{1}{2}$ and $\frac{1}{8}$ for the laminar and the general case, respectively.

Subsequently, we tighten the slackness assumption further to obtain an $\Omega(1)$-approximation algorithm for any constant slackness $\lambda \in (0, 1)$ for the laminar case and any constant $\lambda \in (0, \frac{1}{4})$ for the general case. In the special case where the weight of the job is equal to its area, we extend an algorithm due to Chen, Hassin and Tzur [11] to obtain an $\Omega(1)$-approximation guarantee for the general case with no assumption on slackness.

Our algorithm for the laminar case relies on a non-trivial combination of a *packing* phase and a *scheduling* phase. While the first phase ensures that the output solution has high profit, the second phase guarantees its feasibility. To facilitate a successful completion of the selected jobs, we formulate a set of conditions that must be satisfied in the packing phase. Both phases make use of the structural properties of a *laminar* family of intervals. In the packing phase, we apply our rounding procedure (for the LP solution) to the tree representation of the intervals.[3] We further use this tree in the scheduling phase, to feasibly assign the resource to the selected jobs in a bottom-up fashion. Our framework for solving MaxT is general, and may therefore find use in other settings of *non-consecutive* resource allocation.

---

[2] See the formal definition in Section 2.

[3] This procedure bears some similarity to the *pipage* rounding technique of [2].

For the MinR problem, we obtain (in Section 4) an $O(\log d)$-approximation algorithm for any constant $d \geq 1$, under a mild assumption that any job has a window of size $\Omega(d^2 \log d \log T)$, where $T = \max_j d_j$. We show that this assumption can be removed, leading to a slight degradation in the approximation factor to $O(\log d \log^* T)$, where $\log^* T$ is the smallest integer κ such that $\underbrace{\log \log \ldots \log}_{\kappa \text{ times}} T \leq 1$. Our approach builds on a formulation of the problem as a configuration LP, inspired by the works of [3, 16]. However, we quickly deviate from these prior approaches, in order to handle the time-windows and the extra constraints. Our algorithm involves two main phases: *a maximization phase* and *residual phase*. Roughly speaking, a configuration is a subset of jobs that can be feasibly assigned to a host at a given time slot t. For each t, we choose $O(m \log d)$ configurations with probabilities proportional to their LP-values. In this phase, jobs may be allocated the resource only for part of their processing length. In the second phase, we construct a residual instance based on the amount of time each job has been processed. A key challenge is to show that, for any time window χ, the total "area" of jobs left to be scheduled is at most $1/d$ of the original total area. We use this property to solve the residual instance.

2 Preliminaries

We start with some definitions and notation. For our preemptive variants of RAP, we assume w.l.o.g. that each host has a unit amount of each resource. We further assume that time is slotted. We allow non-consecutive allocation of a resource to each job, as well as job migration. Multiple jobs can be assigned to the same machine at a given time but no job can be processed by multiple machines at the same time. Formally, we denote the set of jobs assigned to host $i \in M$ at time t by $S_{i,t}$. We say that job j is *completed* if there are p_j time slots $t \in [r_j, d_j] = \chi_j$ in which j is allocated its required amount of the resource on some host. A job j is completed if $|\{t \in \chi_j : \exists i \in M \text{ such that } j \in S_{i,t}\}| \geq p_j$. Let $T = \max_{j \in J} d_j$ be the latest due date of any job.

In MaxT, each job $j \in J$ has a resource requirement $s_j \in (0, 1]$. An assignment of a subset of jobs $S \subseteq J$ to the hosts in M is feasible if each job $j \in S$ is completed, and for any time slot t and host $i \in M$, $\sum_{j \in S_{i,t}} s_j \leq 1$, i.e., the sum of requirements of all jobs assigned to host i is at most the available resource. For the MinR variant, we assume multiple resources. Thus, each job j has a resource requirement vector $\bar{s}_j \in [0, 1]^d$, for some constant $d \geq 1$. Further, each host has a unit amount of each of the d resources. An assignment of a set of jobs $S_{i,t}$ to a host $i \in M$ at time t is feasible if $\sum_{j \in S_{i,t}} \bar{s}_j \leq \bar{1}^d$.

Let $a_j = s_j p_j$ denote the total resource requirement (or, *area*) of job $j \in J$ and refer to the quantity w_j/a_j as the *density* of job j. Finally, a set of intervals is *laminar* if for any two intervals χ' and χ'', exactly one of the following holds: $\chi' \subseteq \chi''$, $\chi'' \subset \chi'$ or $\chi' \cap \chi'' = \phi$.

3 Throughput Maximization

We first consider the case where $\mathcal{L} = \{\chi_j : j \in J\}$ forms a *laminar* family of intervals. In Section 3.1, we present an $\Omega(1)$-approximation algorithm for the laminar case when $\lambda \in \left(0, 1 - \frac{2}{m+2}\right)$. Following this, we describe (in Section 3.2) our constant approximation for the general case for $\lambda \in \left(0, \frac{1}{4} - \frac{1}{2(m+2)}\right)$. We then show, in Section 3.3, how to tighten the results to any constant slackness parameter (i) $\lambda \in (0, 1)$ in the laminar case (ii) $\lambda \in (0, \frac{1}{4})$ in the general case. As an interesting corollary, we obtain an $\Omega\left(\frac{1}{\log n}\right)$-approximation algorithm

for the general MaxT problem with no slackness assumption. Further, we show that in the special case of maximum utilization (i.e., the profit of each job equals its "area"), we obtain an $\Omega(1)$ guarantee with no assumption on the slackness.

3.1 The Laminar Case

Our algorithm proceeds in two phases. While the first phase ensures that the output solution has high profit, the second phase guarantees its feasibility. Specifically, let $\omega \in (0, 1 - \frac{\lambda}{m}]$ be a parameter (to be determined).

In Phase 1, we find a subset of jobs S satisfying a knapsack constraint for each χ. Indeed, any feasible solution guarantees that the total area of jobs within any time-window $\chi \in \mathcal{L}$ is at most $m|\chi|$. Our knapsack constraints further restrict the total area of jobs in χ to some fraction of $m|\chi|$. We adopt an LP-rounding based approach to compute a subset S that is optimal subject to the further restricted knapsack constraints. (We remark that a dynamic programming approach would work as well. However, such an approach would not provide us with any intuition as to how an optimal solution for the further restricted instance compares with the optimal solution of the original instance.)

In Phase 2 we allocate the resource to the jobs in S, by considering separately each host i at a given time slot $t \in [T]$ as a unit-sized bin (i, t) and iteratively assigning each job $j \in S$ to a subset of such available bins, until j has the resource allocated for p_j distinct time slots. An outline of the two phases is given in Algorithm 1.

Algorithm 1 Throughput maximization algorithm outline.

Input: Set of jobs J, hosts M and a parameter $\omega \in (0, 1 - \frac{\lambda}{m}]$
Output: Subset of jobs $S \subseteq J$ and a feasible assignment of S to the hosts in M
Phase 1: Select a subset $S \subseteq J$, such that for each $\chi \in \mathcal{L}$:
$\sum_{j \in S : \chi_j \subseteq \chi} a_j \leq (\omega + \frac{\lambda}{m}) m|\chi|$
Phase 2: Find a feasible allocation of the resource to the jobs in S

Phase 1: The algorithm starts by finding a subset of jobs $S \subseteq J$ such that for any $\chi \in \mathcal{L}$: $\sum_{j \in S : \chi_j \subseteq \chi} a_j \leq (\omega + \frac{\lambda}{m}) m|\chi|$. We solve the following LP relaxation, in which we impose stricter constraint on the total area of the jobs assigned in each time window χ.

$$\textbf{LP:} \quad \begin{array}{lll} \text{Maximize} & \sum_{j \in J} w_j x_j & \\ \text{Subject to:} & \sum_{j : \chi_j \subseteq \chi} a_j x_j \leq \omega m|\chi| & \forall \chi \in \mathcal{L} \\ & 0 \leq x_j \leq 1 & \forall j \in J \end{array}$$

Rounding the Fractional Solution: Suppose $\mathbf{x}^* = (x^*_j : j \in J)$ is an optimal fractional solution for the LP. Our goal is to construct an integral solution $\hat{\mathbf{x}} = (\hat{x}_j : j \in J)$. We refer to a job j with $x^*_j \in (0, 1)$ as a *fractional job*, and to the quantity $a_j x^*_j$ as its *fractional area*. W.l.o.g., we may assume that for any interval $\chi \in \mathcal{L}$, there is at most one job j with $\chi_j = \chi$ such that $0 < x^*_j < 1$, i.e., it is fractional. Indeed, if two such jobs exist, then the fractional value of the higher density job (breaking ties arbitrarily) can be increased to obtain a solution no worse than the optimal. Note, however, that there could be fractional jobs j' with $\chi_{j'} \subset \chi$.

We start by setting $\hat{x}_j = x^*_j$ for all $j \in J$. Consider the tree representation of $\mathcal{L}$, which contains a node (also denoted by χ) for each $\chi \in \mathcal{L}$, and an edge between nodes corresponding to χ and χ', where $\chi' \subset \chi$, if there is no interval $\chi'' \in \mathcal{L}$ such that $\chi' \subset \chi'' \subset \chi$.[4] Our

[4] Throughout the discussion we use interchangeably the terms *node* and *interval* when referring to a time-window $\chi \in \mathcal{L}$.

rounding procedure works in a *bottom-up* fashion. As part of this procedure, we label the nodes with one of two possible colors: *gray* and *black*. Initially, all leaf nodes are colored *black*, and all internal nodes are colored *gray*. The procedure terminates when all nodes are colored *black*. A node χ is colored as *black* if the following property holds:

▶ **Property 1.** *For any path $\mathcal{P}(\chi, \chi_l)$ from χ to a leaf χ_l there is at most one fractional job j such that χ_j lies on $\mathcal{P}(\chi, \chi_l)$.*

We note that the property trivially holds for the leaf nodes. Now, consider a *gray* interval χ with children $\chi_1, \chi_2, \ldots, \chi_\nu$, each colored *black*. Note that χ is well defined because leaf intervals are all colored black. If there is no fractional job that has χ as its time-window, Property 1 follows by induction, and we color χ *black*. Assume now that j is a fractional job that has χ as its time-window (i.e., $\chi_j = \chi$). If there is no other fractional job that has its time-window (strictly) contained in χ, Property 1 is trivially satisfied. Therefore, assume that there are other fractional jobs $j_1, j_2, \ldots, j_l$ that have their time-windows (strictly) contained in χ. Now, we decrease the fractional area (i.e., the quantity $a_j \hat{x}_j$) of j by Δ and increase the fractional area of jobs in the set $\{j_1, j_2, \ldots, j_l\}$ by Δ_k for job j_k, such that $\Delta = \sum_{k\in[l]} \Delta_k$. Formally, we set $\hat{x}_j \to \hat{x}_j - \frac{\Delta}{a_j}$ and $\hat{x}_{j_k} \to \hat{x}_{j_k} + \frac{\Delta_k}{a_{j_k}}$. We choose these increments such that either $\hat{x}_j$ becomes 0, or for each $k \in [l]$, $\hat{x}_{j_k}$ becomes 1. Clearly, in both scenarios, Property 1 is satisfied, and we color χ *black*.

When all nodes are colored *black*, we round up the remaining fractional jobs. Namely, for all jobs j such that $\hat{x}_j \in (0, 1)$, we set $\hat{x}_j = 1$. It is important to note that by doing so we may violate the knapsack constraints. However, in Theorem 1, we bound the violation.

▶ **Theorem 1.** *Suppose $\mathcal{I} = (J, M, \mathcal{L})$ is a laminar instance of* MaxT *with optimal profit W and $\forall j \in J$: $p_j \le \lambda|\chi_j|$. For any $\omega \in (0, 1 - \frac{\lambda}{m}]$, the subset $S = \{j \in J : \hat{x}_j = 1\}$, obtained as above, satisfies $\sum_{j\in S} w_j \ge \omega W$, and for any $\chi \in \mathcal{L}$, $\sum_{j\in S:\chi_j\subseteq\chi} a_j \le (\omega + \frac{\lambda}{m})m|\chi|$.*

Proof. We first observe that any optimal solution $\mathbf{x}^*$ for the LP satisfies: $\sum_{j\in J} w_j x^*_j \ge \omega W$. Indeed, consider an optimal solution O for the instance $\mathcal{I}$. We can construct a fractional feasible solution $\mathbf{x}'$ for the LP by setting $x'_j = \omega$ if $j \in O$; otherwise, $x'_j = 0$. Clearly, $\mathbf{x}'$ is a feasible solution for the LP with profit ωW.

Consider an integral solution $\hat{\mathbf{x}}$, obtained by applying the rounding procedure on $\mathbf{x}^*$. We first show that $\sum_{j\in J} w_j \hat{x}_j \ge \omega W$. To this end, we prove that $\sum_{j\in J} w_j \hat{x}_j \ge \sum_{j\in J} w_j x^*_j \ge \omega W$. Suppose we decrease the fractional area of a job j by an amount Δ, i.e., we set $\hat{x}_j \leftarrow \hat{x}_j - \frac{\Delta}{a_j}$. By the virtue of our procedure, we must simultaneously increase the fractional area of some subset of jobs F_j, where for each $k \in F_j$ we have $\chi_k \subset \chi_j$. Further, the combined increase in the fractional area of the jobs in F_j is the same Δ. Now, we observe that the density of job j (i.e., $\frac{w_j}{a_j}$) cannot be higher than any of the jobs in F_j. Indeed, if $j' \in F_j$ has density strictly lower than j, then the optimal solution $\mathbf{x}^*$ can be improved by decreasing the fractional area of j' by some ϵ while increasing that of j by the same amount (it is easy to see that no constraint is violated in this process) – a contradiction. Therefore, our rounding procedure will never result in a loss, and $\sum_{j\in J} w_j \hat{x}_j \ge \sum_{j\in J} w_j x^*_j \ge \omega W$.

We now show that, for each $\chi \in \mathcal{L}$, $\sum_{j\in J:\chi_j\subseteq\chi} a_j \hat{x}_j \le (\omega + \frac{\lambda}{m})m|\chi|$. First, observe that for any *gray* interval χ the total fractional area is *conserved*. This is true because there is no transfer of fractional area from the subtree rooted at χ to a node outside this subtree until χ is colored black. Now, consider an interval χ that is colored *black*. We note that for any job j with $x^*_j = 0$, our algorithm ensures that $\hat{x}_j = 0$, i.e., it creates no new fractional jobs. Consider the vector $\hat{\mathbf{x}}$ when the interval χ is converted from *gray* to *black*. At this stage, we have that the total (fractional) area packed in the subtree rooted at χ is

$V(\chi) \stackrel{def}{=} \sum_{j\in J:\chi_j\subseteq\chi} a_j\hat{x}_j \leq \omega m|\chi|$. Let $\mathcal{F}(\chi)$ denote the set of all fractional jobs j' that have their time-windows contained in χ (i.e., $\chi_{j'} \subseteq \chi$). We claim that the maximum increase in $V(\chi)$ by the end of the rounding procedure is at most $\sum_{j'\in\mathcal{F}(\chi)} a_{j'}$. This holds since our procedure does not change the variables $\hat{x}_j \in \{0,1\}$. Thus, the maximum increase in the total area occurs due to rounding all fractional jobs into complete ones, after all nodes are colored black. To complete the proof, we now show that the total area of the fractional jobs in the subtree rooted at χ satisfies $\mathcal{A}[\chi] \stackrel{def}{=} \sum_{j'\in\mathcal{F}(\chi)} a_{j'} \leq \lambda|\chi|$. We prove this by induction on the level of node χ. Clearly, if χ is a leaf then the claim holds, since there can exist at most one fractional job j in χ, and $a_j \leq p_j \leq \lambda|\chi|$. Suppose that $\{\chi_1, \chi_2, \ldots \chi_l\}$ are the children of χ. If there is a fractional job j with $\chi_j = \chi$ then, by Property 1, there are no other fractional jobs with time-windows contained in χ. Hence, $\mathcal{A}[\chi] = a_j \leq \lambda|\chi|$. Suppose there is no fractional job with $\chi_j = \chi$; then, by the induction hypothesis: $\mathcal{A}[\chi_k] \leq \lambda|\chi_k|$ for all $k \in [l]$. Further, $\sum_{k\in[l]} |\chi_k| \leq |\chi|$ and $\mathcal{A}[\chi] = \sum_{k\in[l]} \mathcal{A}[\chi_k] \leq \sum_{k\in[l]} \lambda|\chi_k| \leq \lambda|\chi|$. ◀

Let O be an optimal solution for $\mathcal{I}$ satisfying: $\forall\chi \in \mathcal{L} \ : \ \sum_{j\in O:\chi_j\subseteq\chi} a_j \leq cm|\chi|$, for some $c \geq 1$. Then it is easy to verify that any optimal solution $\mathbf{x}^*$ for the LP satisfies: $\sum_{j\in J} w_j x^*_j \geq \frac{\omega}{c}W$. Hence, we have

▶ **Corollary 2.** *Suppose $\mathcal{I} = (J, M, \mathcal{L})$ is a laminar instance of* MaxT*, such that $\forall j \in J : p_j \leq \lambda|\chi_j|$. Let $S^+ \subseteq J$ be a subset of jobs of total profit W satisfying $\forall\chi \in \mathcal{L}$: $\sum_{j\in S^+:\chi_j\subseteq\chi} a_j \leq cm|\chi|$, for some $c \geq 1$. Then, for any $\omega \in (0, 1 - \frac{\lambda}{m}]$, there exists a subset $S \subseteq J$ satisfying $\sum_{j\in S} w_j \geq \frac{\omega}{c}W$, such that $\forall\chi \in \mathcal{L}$, $\sum_{j\in S:\chi_j\subseteq\chi} a_j \leq (\omega + \frac{\lambda}{m})m|\chi|$.*

Phase 2: In Phase 1 we obtained a subset $S \subseteq J$, such that for each $\chi \in \mathcal{L}$: $\sum_{j\in S:\chi_j\subseteq\chi} a_j \leq (\omega + \frac{\lambda}{m})m|\chi|$. We now show that it is always possible to find a feasible packing of all jobs in S. We refer to host i at time t as a *bin* (i, t). In the allocation phase we label a *bin* with one of three possible colors: *white*, *gray* or *black*. Initially, all bins are colored *white*. We color a bin (i, t) gray when some job j is assigned to host i at time t and color it black when we decide to assign no more jobs to this bin. Our algorithm works in a bottom-up fashion and marks an interval χ as *done* when it has successfully completed all the jobs j with $\chi_j \subseteq \chi$. Consider an interval χ such that any $\chi' \subset \chi$ has already been marked *done*. Let $j \in S$ be a job with time-window $\chi_j = \chi$, that has not been processed yet. To complete job j, we must pick p_j distinct time slots in χ and assign it to a bin in each slot. Suppose that we have already assigned the job to $p'_j < p_j$ slots so far. Denote by $avail(j) \subseteq \chi$ the subset of time slots where j has not been assigned yet. We pick the next slot and bin as shown in Algorithm 2.

▶ **Theorem 3.** *For any $\lambda < 1 - \frac{2}{m+2}$, there exists a $\frac{1}{2} - \lambda\left(\frac{1}{2} + \frac{1}{m}\right)$-approximation algorithm for the laminar* MaxT *problem, assuming that $p_j \leq \lambda|\chi_j|$ for all $j \in J$.*

Proof. Given an instance $\mathcal{I} = (J, M, \mathcal{L})$ and a parameter $\omega \in (0, 1 - \frac{\lambda}{m}]$, let W denote the optimal profit. We apply Theorem 1 to find a subset of jobs $S \subseteq J$ of profit ωW, such that for any $\chi \in \mathcal{L}$: $\sum_{j\in S:\chi_j\subseteq\chi} a_j \leq (\omega + \frac{\lambda}{m})m|\chi|$. We now show that there is a feasible resource assignment to the jobs in S for $\omega = \frac{1}{2} - \lambda\left(\frac{1}{2} + \frac{1}{m}\right)$. Clearly, this would imply the theorem.

We show that for the above value of ω Algorithm 2 never reports **fail**, i.e., the resource is feasibly allocated to all jobs in S. Assume towards contradiction that Algorithm 2 reports **fail** while assigning job j. Suppose that j was assigned to $p'_j < p_j$ bins before this **fail**. For $t \in \chi = \chi_j$, we say that bin (i, t) is *bad* if either (i, t) is colored gray, or j has been assigned to some bin (i', t) in the same time slot. We first show that the following invariant holds, as long as no job j^+ such that $\chi \subset \chi_{j^+}$ has been allocated the resource: the number of bad bins

Algorithm 2 Resource allocation to job j in a single time slot.

if there exists a gray bin (i,t) in $avail(j)$ **then**
  let $S_{(i,t)}$ be the set of jobs assigned to this bin
  if $\sum_{j' \in S_{(i,t)}} s_{j'} + s_j \leq 1$ **then**
    assign j to host i at time t
  else if there exists a white bin (i',t') in $avail(j)$ **then**
    assign j to host i' at time t'.
    color (i,t) and (i',t') black
    pair up $(i,t) \leftrightarrow (i',t')$
  else
    report **fail**
  end if
else if there exists a white bin (i,t) in $avail(j)$ **then**
  assign j to host i at time t
  color the bin (i,t) gray
else
  report **fail**
end if

while processing job j is at most $\lambda m|\chi|$. Assuming that the claim is true in each of the child intervals of χ, $\{\chi_1, \chi_2 \ldots, \chi_l\}$, before any job with time window χ is allocated the resource, we have the number of bad bins = number of gray bins is at most $\sum_{k\in[l]} \lambda m|\chi_k| \leq \lambda m|\chi|$. Now, consider the iteration in which we assign j to host i at time t. If (i,t) is a gray bin, then the number of bad bins cannot increase. On the other hand, suppose (i,t) was white before we assign j. If there are no gray bins in χ, then the number of bad bins is at most $mp_j \leq \lambda m|\chi|$. Suppose there exist some gray bins, and consider those bins of the form (i',t') such that job j has not been assigned to any host at time t'. If there are no such bins, then again the number of bad bins is at most $mp_j \leq \lambda m|\chi|$. Otherwise, we must have considered one such gray bin (i',t') and failed to assign j to host i' at time t'. By the virtue of the algorithm, we must have colored both (i,t) and (i',t') black. Thus, the number of bad bins does not increase, and our claim holds. Now, since we pair the black bins $(i,t) \leftrightarrow (i',t')$ only if $\sum_{j\in S_{(i,t)}} s_j + \sum_{j'\in S_{(i',t')}} s_{j'} > 1$, the total number of black bins $< 2(\omega + \frac{\lambda}{m})m|\chi|$. Hence, the total number of bins that are black or bad is $< (\lambda + 2(\omega + \frac{\lambda}{m}))m|\chi|$. Now, setting $\omega = \frac{1}{2} - \lambda\left(\frac{1}{2} + \frac{1}{m}\right)$, there should be at least one bin (i^*,t^*) that is neither black nor bad. But in this case, we could have assigned j to host i^* at time t^*, which is a contradiction to the assumption that the algorithm reports a **fail**. ◀

For convenience, we restate the claim shown in the proof of Theorem 3.

▶ **Corollary 4.** *Let $\mathcal{I} = (J, M, \mathcal{L})$ be a laminar instance where $p_j \leq \lambda|\chi_j|$ $\forall j \in J$, for $\lambda \in (0,1)$. Let $S \subseteq J$ be a subset of jobs, such that for any $\chi \in \mathcal{L}$: $\sum_{j\in S:\chi_j\subseteq\chi} a_j \leq (\omega + \frac{\lambda}{m})m|\chi|$, where $\omega \leq \frac{1}{2} - \lambda\left(\frac{1}{2} + \frac{1}{m}\right)$. Then, there exists a feasible resource assignment to the jobs in S.*

3.2 The General Case

We use a simple transformation of general instances of MaxT into laminar instances and prove an $\Omega(1)$-approximation guarantee. Let $\mathcal{W}$ denote the set of all time-windows for jobs in J, i.e., $\mathcal{W} = \{\chi_j : j \in J\}$. We now construct a laminar set of intervals $\mathcal{L}$ and a mapping $\mathfrak{L} : \mathcal{W} \to \mathcal{L}$. Recall that $T = \max_{j\in J} d_j$. The construction is done via a binary tree $\mathcal{T}$ whose nodes correspond to intervals $[l,r] \subseteq [T]$. The construction is described in Algorithm 3.

Algorithm 3 Transformation into a laminar set.

Input: Job set J and $\mathcal{W} = \{\chi_j : j \in J\}$
Output: Laminar set of intervals $\mathcal{L}$ and a mapping $\mathfrak{L} : \mathcal{W} \to \mathcal{L}$

1: let $[T]$ be the root node of tree $\mathcal{T}$
2: **while** $\exists$ a leaf node $[l, r]$ in $\mathcal{T}$ such that $r - l > 1$ **do**
3: add to $\mathcal{T}$ two nodes $[l, \lfloor \frac{l+r}{2} \rfloor]$ and $[\lfloor \frac{l+r}{2} \rfloor + 1, r]$ as the children of $[l, r]$
4: **end while**
5: let $\mathcal{L}$ be the set of intervals corresponding to the nodes of $\mathcal{T}$
6: For each $\chi \in \mathcal{W}$, let $\mathfrak{L}(\chi) = \chi'$, where χ' is the largest interval in $\mathcal{L}$ contained in χ, breaking ties by picking the *rightmost* interval.

▶ **Lemma 5.** *In Algorithm 3, the following properties hold (Proof in final version):*

1. For any $j \in J$, $|\chi_j| \leq 4|\mathfrak{L}(\chi_j)|$.

2. For $\chi \in \mathcal{L}$, let $\tilde{\chi} = \{t \in \chi_j : j \in J, \mathfrak{L}(\chi_j) = \chi\}$, i.e., the union of all time-windows in $\mathcal{W}$ that are mapped to χ. Then, $|\tilde{\chi}| \leq 4|\chi|$.

▶ **Theorem 6.** *For any $\lambda < \frac{1}{4} - \frac{1}{2(m+2)}$, there exists a $\frac{1}{8} - \lambda\left(\frac{1}{2} + \frac{1}{m}\right)$-approximation algorithm for* MaxT *, assuming that $p_j \leq \lambda|\chi_j|$ for all $j \in J$.*

Proof. Given an instance $(J, M, \mathcal{W})$ of MaxT with slackness parameter $\lambda \in (0, 1)$, we first use Algorithm 3 to obtain a laminar set of intervals $\mathcal{L}$ and the corresponding mapping $\mathfrak{L} : \mathcal{W} \to \mathcal{L}$. Consider a new laminar instance $(J_\ell = \{j_\ell : j \in J\}, M_\ell = M, \mathcal{L})$, constructed by setting $\chi_{j_\ell} = \mathfrak{L}(\chi_j)$. Note that if $S_\ell \subseteq J_\ell$ is a feasible solution for this new instance, the corresponding set $S = \{j : j_\ell \in S_\ell\}$ is a feasible solution for the original instance. Let λ_ℓ denote the slackness parameter for the new instance. We claim that $\lambda_\ell \leq 4\lambda$. Assume this is not true, i.e., there exists a job j_ℓ, such that $p_{j_\ell} > 4\lambda|\chi_{j_\ell}|$; however, by Lemma 5, we have $p_{j_\ell} = p_j \leq \lambda|\chi_j| \leq 4\lambda|\chi_{j_\ell}|$. A contradiction. Now, suppose $O \subseteq J$ is an optimal solution of total profit W for the original (non-laminar) instance. Consider the corresponding subset of jobs $O_\ell = \{j_\ell : j \in O\}$. By Lemma 5, for any $\chi \in \mathcal{L}$, $|\tilde{\chi}| \leq 4|\chi|$. It follows that, for any $\chi \in \mathcal{L}$, $\sum_{j_\ell \in O_\ell : \chi_{j_\ell} \subseteq \chi} a_{j_\ell} = \sum_{j \in O : \mathfrak{L}(\chi_j) \subseteq \chi} a_j \leq 4m|\chi|$. Now, we use Corollary 2 for the laminar instance, taking $c = 4$, $S^+ = O_\ell$ and $\lambda_\ell \in (0, 1)$. Then, for any $\omega \in (0, 1 - \frac{\lambda_\ell}{m}]$, there exists $S_\ell \subseteq J_\ell$ of total profit $\sum_{j_\ell \in S_\ell} w_j \geq \frac{\omega}{c} W$, such that $\forall \chi \in \mathcal{L}$, $\sum_{j_\ell \in S_\ell : \chi_j \subseteq \chi} a_{j_\ell} \leq (\omega + \frac{\lambda_\ell}{m})m|\chi|$. By Corollary 4, there is a feasible assignment of the resource to the jobs in S_ℓ for $\omega \leq \frac{1}{2} - \lambda_\ell\left(\frac{1}{2} + \frac{1}{m}\right)$. Taking $\omega = \dfrac{1}{2} - 4\lambda\left(\dfrac{1}{2} + \dfrac{1}{m}\right) \leq \dfrac{1}{2} - \lambda_\ell\left(\dfrac{1}{2} + \dfrac{1}{m}\right)$, we have the approximation ratio $\frac{w}{c} = \frac{1}{8} - 4\lambda\left(\frac{1}{8} + \frac{1}{4m}\right)$, for any $\lambda < \frac{1}{4} - \frac{1}{2(m+2)}$. We now return to the original instance and take for the solution the set $S = \{j : j_\ell \in S_\ell\}$. ◀

3.3 Eliminating the Slackness Requirements

In this section we show that the slackness requirements in Theorems 3 and 6 can be eliminated, while maintaining a constant approximation ratio for MaxT . In particular, for laminar instances, we show below that Algorithm 1 can be used to obtain a polynomial time $\Omega(1)$-approximation for any constant slackness parameter $\lambda \in (0, 1)$. For general MaxT instances, this leads to an $\Omega(1)$-approximation for any constant $\lambda \in (0, \frac{1}{4})$. We also show a polynomial time $\Omega(\frac{1}{\log n})$-approximation algorithm for general MaxT using no assumption on slackness. We use below the next result, for instances with "large" resource requirement. The proof is deferred to the full version.

▶ **Lemma 7.** *For any $\delta \in (0,1)$ there is an $\Omega(\frac{1}{\log(1/\delta)})$-approximation for any instance $\mathcal{I} = (J, M, \mathcal{W})$ of MaxT satisfying $s_j \geq \delta \ \forall\ j \in J$.*

3.3.1 Laminar Instances

Recall that $m = |M|$ is the number of hosts. Given a fixed $\lambda \in (0,1)$, let

$$\alpha = \alpha(m, \lambda) = \frac{\lambda(1-\lambda)}{1-\lambda+\frac{\lambda}{m}}. \tag{1}$$

In Phase 1 of Algorithm 1, we round the LP solution to obtain a subset of jobs $S \subseteq J$. We first prove the following.

▶ **Lemma 8.** *Let $\lambda \in (0,1)$ be a slackness parameter, and*

$$\omega = (1-\alpha)(1-\lambda) - \frac{\alpha\lambda}{m}, \tag{2}$$

where α is defined in (1). Then, given a laminar instance $\mathcal{I} = (J, M, \mathcal{L})$ satisfying $p_j \leq \lambda|\chi_j|$ and $s_j \leq \alpha$, there is a feasible allocation of the resource to the jobs in S.

Proof. We generate a feasible schedule of the jobs in S proceeding bottom-up in each laminar tree. That is, we start handling job j only once all the jobs ℓ with time windows $\chi_\ell \subset \chi_j$ have been scheduled. Jobs having the same time window are scheduled in an arbitrary order. Let j be the next job, whose time window is $\chi = \chi_j$. We can view the interval χ as a set of $|\chi|$ time slots, each consisting of m unit size bins. We say that a time slot $t \in \chi_j$ is "bad" for job j if there is no space for one processing unit of j (i.e., an "item" of size s_j) in any of the bins in t; else, time slot t is "good". We note that immediately before we start scheduling job j the number of bad time slots for j is at most $\frac{m|\chi|(1-\lambda)(1-\alpha)-a_j}{m(1-s_j)}$. Indeed, by Theorem 1, choosing for ω the value in (2), after rounding the LP solution the total area of jobs $\ell \in S$, such that $\chi_\ell \subseteq \chi_j$, is at most

$$(\omega + \frac{\alpha\lambda}{m})m|\chi| = ((1-\alpha)(1-\lambda) - \frac{\alpha\lambda}{m} + \frac{\alpha\lambda}{m})m|\chi|. \tag{3}$$

In addition, for a time slot t to be "bad" for job j, each bin in t has to be at least $(1-s_j)$-full. Hence, the number of good time slots for j is at least

$$\begin{aligned} |\chi| - \frac{m|\chi|(1-\lambda)(1-\alpha)-a_j}{m(1-s_j)} &= |\chi|(1 - \frac{(1-\lambda)(1-\alpha)}{1-s_j}) + \frac{a_j}{m(1-s_j)} \\ &\geq \frac{p_j}{\lambda}(1 - \frac{(1-\lambda)(1-\alpha)}{1-s_j}) + \frac{a_j}{m(1-s_j)} \geq p_j \end{aligned}$$

The first inequality follows from the fact that $p_j \leq \lambda|\chi_j| = \lambda|\chi|$, and the second inequality holds since $s_j \leq \alpha$. Hence, job j can be feasibly scheduled, for any $j \in S$. ◀

Using Lemmas 7 and 8, we prove our main result.

▶ **Theorem 9.** *For any $m \geq 1$ and constant $\lambda \in (0,1)$, MaxT admits a polynomial time $\Omega(1)$-approximation on any laminar instance $\mathcal{I} = (J, M, \mathcal{L})$ with slackness parameter λ.*

Proof. Given a laminar instance $\mathcal{I}$ satisfying the slackness condition, we handle separately two subsets of jobs.

Subset 1: Jobs j satisfying $s_j \leq \alpha = \alpha(m, \lambda)$, where α is defined in (1). We solve MaxT for these jobs using Algorithm 1, taking the value of ω as in (2). By Theorem 1, the approximation ratio is $\omega = (1-\alpha)(1-\lambda) - \frac{\alpha\lambda}{m} = (1-\lambda)^2$, i.e., we have a constant factor.

Subset 2: For jobs j satisfying $s_j > \alpha$, use Lemma 7 to obtain an $\Omega(\frac{1}{\log(1/\alpha)})$-approximation.

Taking the best among the solutions for the two subsets of jobs, we obtain an $\Omega(1)$-approximation. ◀

3.3.2 The General Case

Recall that, given a general MaxT instance, $(J, M, \mathcal{W})$, with a slackness parameter $\lambda \in (0, 1)$, our transformation yields a new laminar instance $(J_\ell = \{j_\ell : j \in J\}, M_\ell = M, \mathcal{L})$ with a slackness parameter $\lambda_\ell \leq 4\lambda$ (see the proof of Theorem 6). Now, define

$$\alpha_\ell = \alpha_\ell(m, \lambda_\ell) = \frac{\lambda_\ell(1-\lambda_\ell)}{1 - \lambda_\ell + \frac{\lambda_\ell}{m}}, \tag{4}$$

and set

$$\omega = (1-\alpha_\ell)(1-\lambda_\ell) - \frac{\alpha_\ell\lambda_\ell}{m}. \tag{5}$$

Then, by Lemma 8, we have that any job $j_\ell \in J_\ell$ selected for the solution set S can be assigned the resource (using Algorithm 1).

▶ **Theorem 10.** *For any $m \geq 1$ and constant $\lambda \in (0, \frac{1}{4})$, MaxT admits a polynomial time $\Omega(1)$-approximation on any instance $\mathcal{I} = (J, M, \mathcal{W})$ with slackness parameter λ.*

Proof. Given such an instance $\mathcal{I}$, consider the resulting laminar instance. As before, we handle separately two subsets of jobs.

Subset 1: For jobs $j_\ell \in J_\ell$ satisfying $s_{j_\ell} \leq \alpha_\ell$, where α_ℓ is defined in (4), apply Algorithm 1 with ω value as in (5). Then, the approximation ratio is $\omega = (1-\lambda_\ell)^2 \geq (1-4\lambda)^2$.

Subset 2: For jobs j_ℓ where $s_{j_\ell} > \alpha_\ell$, use Lemma 7 to obtain $\Omega(\frac{1}{\log(1/\alpha_\ell)})$-approximation.

Taking the best among the solutions for the two subsets of jobs, we obtain an $\Omega(1)$-approximation. ◀

Finally, consider a general instance of MaxT . By selecting $\delta = \frac{1}{n}$, we can apply Lemma 7 to obtain an $\Omega(\frac{1}{\log n})$-approximate solution, S_1 for the jobs $j \in J$ of heights $s_j \geq \frac{1}{n}$. Let S_2 be a solution consisting of all jobs j for which $s_j < \frac{1}{n}$. Note that this solution is feasible since $\sum_{j \in S_2} s_j < 1$. Selecting the highest profit solution between S_1 and S_2, we have:

▶ **Corollary 11.** *There is a polynomial time $\Omega(\frac{1}{\log n})$-approximation algorithm for MaxT .*

3.3.3 Maximizing Utilization

Consider instances of MaxT where the profit gained from scheduling job j is $w_j = a_j = s_j p_j$. We give the proof of the following in [27].

▶ **Theorem 12.** *There is a polynomial time $\Omega(1)$-approximation for any instance of MaxT where $w_j = a_j$ for all $j \in J$.*

4 Resource Minimization

In this section, we consider the MinR problem with d resources, where $d \geq 1$ is some constant. We show that the problem admits an $O(\log d)$-approximation algorithm under some mild assumptions on the slack and minimum window size. Further, we show that the latter assumption can be removed with a slight degradation in the approximation guarantee.

Our approach builds on a formulation of the problem as a configuration LP and involves two main phases: *a maximization phase* and *residual phase*. We start by describing the configuration LP that is at the heart of our algorithm. Let $J_t \subseteq J$ denote the set of all jobs j such that $t \in \chi_j$, i.e., j can be allocated resources at time slot t. For any $t \in [T]$ and $S \subseteq J_t$, $C = (S, t)$ is a valid *configuration* on a single host if $\sum_{j \in S} \bar{s}_j \leq \bar{1}^d$, i.e., the jobs in S can be feasibly assigned to a single host at time slot t. Denote the set of all valid configurations at time t by $\mathcal{C}_t$, and by $\mathcal{C}^j$ the set of all valid configurations (S, t), such that S contains job j. Denote by x_C the indicator variable for choosing configuration C and by m the number of hosts needed to schedule all jobs. The fractional relaxation of the integer program formulation of our problem is given below.

$$
\begin{array}{lll}
\textbf{Primal}: \text{ Minimize} & m & \\
\text{Subject to:} & m - \sum_{C \in \mathcal{C}_t} x_C \geq 0, & \forall t \in [T] \\
& -\sum_{C \in \mathcal{C}^j \cap \mathcal{C}_t} x_C \geq -1, & \forall j \in J,\ t \in [T] \\
& \sum_{C \in \mathcal{C}^j} x_C \geq p_j, & \forall j \in J \\
& x_C \geq 0, & \forall C
\end{array}
$$

The first constraint ensures that we do not pick more than m configurations for each time slot $t \in [T]$. The second constraint guarantees that at most one configuration is chosen for each job j at a given time t. Finally, the last constraint guarantees that each job j is allocated the resource for p_j time slots, i.e., job j is completed. The proof of the following theorem (see [27]) is similar to a result of Fleischer et al. [16], with some differences due to "negative" terms in the objective of the dual program.

▶ **Theorem 13.** *For any $\epsilon > 0$, there is a polynomial time algorithm that yields a $(1+\epsilon)$-approximate solution for the configuration LP.*

Given the objective value m of the approximate LP solution, we choose for each t $O(m \log d)$ configurations with probabilities proportional to their LP-values. In this phase, jobs may be allocated the resource only for part of their processing length. In the second phase, we construct a residual instance based on the amount of time each job has been processed. A key challenge is to show that, for any time window χ, the total "area" of jobs left to be scheduled is at most $1/d$ of the original total area. We use this property to solve the residual instance. The detailed algorithm and its analysis are given in [27].

▶ **Theorem 14.** *Let $(J, \mathcal{W})$ be an instance of* MinR *with slackness parameter $\lambda \in (0, \frac{1}{4})$. Fix an $\epsilon \in (0, 1)$. If $|\chi_j| \geq \frac{1}{m}\theta d^2 \log d \log(T\epsilon^{-\frac{1}{2}})\ \forall\ j \in J$, for sufficiently large constant θ, we obtain an $O(\log d)$ approximation guarantee with probability at least $1 - \epsilon$.*

▶ **Theorem 15.** *Let $(J, \mathcal{W})$ be an instance of* MinR *with slackness parameter $\lambda \in (0, \frac{1}{4})$. Fix an $\epsilon \in (0, 1)$. There is a polynomial time algorithm that yields an $O(\log d \log^* T)$ approximation ratio with probability at least $1 - \epsilon$.*

References

1 Micah Adler, Phillip B Gibbons, and Yossi Matias. Scheduling space-sharing for internet advertising. *Journal of Scheduling*, 5(2):103–119, 2002.

2 Alexander A. Ageev and Maxim Sviridenko. Pipage Rounding: A New Method of Constructing Algorithms with Proven Performance Guarantee. *J. Comb. Optim.*, 8(3):307–328, 2004.

3 Nikhil Bansal, Alberto Caprara, and Maxim Sviridenko. A New Approximation Method for Set Covering Problems, with Applications to Multidimensional Bin Packing. *SIAM J. Comput.*, 39(4):1256–1278, 2009.

4 Nikhil Bansal, Marek Eliáš, and Arindam Khan. Improved approximation for vector bin packing. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1561–1579, 2016.

5 Nikhil Bansal, Zachary Friggstad, Rohit Khandekar, and Mohammad R Salavatipour. A logarithmic approximation for unsplittable flow on line graphs. *ACM Transactions on Algorithms*, 10(1):1, 2014.

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

7 Amotz Bar-Noy, Sudipto Guha, Joseph Naor, and Baruch Schieber. Approximating the Throughput of Multiple Machines in Real-Time Scheduling. *SIAM J. Comput.*, 31(2):331–352, 2001.

8 Gruia Călinescu, Amit Chakrabarti, Howard J. Karloff, and Yuval Rabani. An improved approximation algorithm for resource allocation. *ACM Trans. Algorithms*, 7(4):48:1–48:7, 2011.

9 Venkatesan T Chakaravarthy, Anamitra R Choudhury, Shalmoli Gupta, Sambuddha Roy, and Yogish Sabharwal. Improved algorithms for resource allocation under varying capacity. In *European Symposium on Algorithms*, pages 222–234. Springer, 2014.

10 Chandra Chekuri and Sanjeev Khanna. On multidimensional packing problems. *SIAM journal on computing*, 33(4):837–851, 2004.

11 Bo Chen, Refael Hassin, and Michal Tzur. Allocation of bandwidth and storage. *IIE Transactions*, 34(5):501–507, 2002.

12 Julia Chuzhoy and Paolo Codenotti. Resource minimization job scheduling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 70–83. Springer, 2009.

13 Julia Chuzhoy, Sudipto Guha, Sanjeev Khanna, and Joseph Naor. Machine minimization for scheduling jobs with interval constraints. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 81–90, 2004.

14 Milind Dawande, Subodha Kumar, and Chelliah Sriskandarajah. Performance bounds of algorithms for scheduling advertisements on a web page. *Journal of Scheduling*, 6(4):373–394, 2003.

15 Milind Dawande, Subodha Kumar, and Chelliah Sriskandarajah. Scheduling web advertisements: a note on the minspace problem. *Journal of Scheduling*, 8(1):97–106, 2005.

16 L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko. Tight Approximation Algorithms for Maximum Separable Assignment Problems. *Math. Oper. Res.*, 36(3):416–431, 2011.

17 Kyle Fox and Madhukar Korupolu. Weighted flowtime on capacitated machines. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 129–143. SIAM, 2013.

18 Ari Freund and Joseph Naor. Approximating the advertisement placement problem. *Journal of Scheduling*, 7(5):365–374, 2004.

19 Navendu Jain, Ishai Menache, Joseph Naor, and Jonathan Yaniv. Near-optimal scheduling mechanisms for deadline-sensitive jobs in large computing clusters. *ACM Transactions on Parallel Computing*, 2(1):3, 2015.

20 Klaus Jansen and Lorant Porkolab. On preemptive resource constrained scheduling: polynomial-time approximation schemes. *Integer Programming and Combinatorial Optimization*, pages 329–349, 2002.

21 Bala Kalyanasundaram and Kirk Pruhs. Eliminating Migration in Multi-processor Scheduling. *J. Algorithms*, 38(1):2–24, 2001.

22 Arshia Kaul, Sugandha Aggarwal, Anshu Gupta, Niraj Dayama, Mohan Krishnamoorthy, and PC Jha. Optimal advertising on a two-dimensional web banner. *International Journal of System Assurance Engineering and Management*, pages 1–6, 2017.

23 Subodha Kumar, Milind Dawande, and Vijay Mookerjee. Optimal scheduling and placement of internet banner advertisements. *IEEE Transactions on Knowledge and Data Engineering*, 19(11), 2007.

24 Eugene L Lawler. A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Annals of Operations Research*, 26(1):125–133, 1990.

25 Shinjini Pandey, Goutam Dutta, and Harit Joshi. Survey on Revenue Management in Media and Broadcasting. *Interfaces*, 47(3):195–213, 2017.

26 Cynthia A. Phillips, R. N. Uma, and Joel Wein. Off-line admission control for general scheduling problems. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 879–888, 2000.

27 Kanthi K. Sarpatwar, Baruch Schieber, and Hadas Shachnai. The Preemptive Resource Allocation Problem. *CoRR*, abs/1811.07413, 2018. `arXiv:1811.07413`.