Scheduling Resources for Throughput Maximization

Venkatesan T. Chakaravarthy[1], Amit Kumar[2], Vinayaka Pandit[1], Sambuddha Roy[1], and Yogish Sabharwal[1]

[1] IBM Research - India
{vechakra, pvinayak, sambuddha, ysabharwal}@in.ibm.com
[2] Indian Institute of Technology - Delhi
amitk@cse.iitd.ac.in

April 15, 2011

Abstract

We consider the problem of scheduling a set of resources over time. Each resource is specified by a set of time intervals (and the associated amount of resource available), and we can choose to schedule it in one of these intervals. The goal is to maximize the number of demands satisfied, where each demand is an interval with a starting and ending time, and a certain resource requirement. This problem arises naturally in many scenarios, e.g., the resource could be an energy source, and we would like to suitably combine different energy sources to satisfy as many demands as possible. Another example of resource is network bandwidth, where we would like to maximize the number of demands routed given that we have bought *chunks* of bandwidth which can be used in a flexible manner over time. We give a constant factor randomized approximation algorithm for this problem, under suitable assumptions (the so called no-bottleneck assumptions). We show that without these assumptions, the problem is as hard as the independent set problem. Our proof requires a novel configuration LP relaxation for this problem. The LP relaxation exploits the pattern of demand sharing that can occur across different resources.

1 Introduction

We consider the problem of scheduling jobs when the resources required for executing the jobs have limited availability. We use the terms "resource" and "machine" interchangeably. In the scheduling literature it is typical to assume that the machines are always available for scheduling and the goal is to schedule jobs in a manner that satisfies all the constraints of correct scheduling and optimize some objective function like makespan, flowtime, completion time, etc. This is in stark contrast to our scenario, where the machines are not available at all the times. Each machine specifies a time window within which it is available, along with a duration for which the machine can be used for. We generalize such a setting by allowing each machine to specify a list of intervals when it is available and the scheduler can pick only one interval for each machine. Therefore, the main challenge is to judiciously schedule the machines in one of their allowed intervals so as to maximize the number of jobs that can be executed in the chosen intervals.

The setting of limited machine (or resource) availability considered in this paper arises naturally in several scenarios. As an example, consider a workforce management scenario, where the

employees specify different intervals of the day when they are available and the scheduler can only pick one of their intervals (an employee can have only a single shift in a day). The goal of the scheduler is to pick a shift for each employee such that a maximum number of jobs can be processed (jobs are specified by intervals). Another scenario that is gaining importance is the domain of proprietary microgrids where the intermittent nature of the power sources poses a significant scheduling issue [14]. The proprietary microgrids use traditional energy sources as well as non-traditional energy sources such as wind and solar energy. Typically, the non-traditional energy source is harnessed when available and charged into a temporary storage (such as batteries). The energy stored in a battery can last only for a limited period of time. So, a solar energy source may be specified by the different periods of the day when it is available and the duration for which it can be used is limited by the battery life. The factory has a knowledge of all the jobs it needs to execute and the intervals in which they need to be executed. It would like to optimally schedule the power sources in its grid so as to maximize the number of jobs that can be finished.

Problem Definition: We define the problem of *resource scheduling for throughput maximization* (RSTM) as follows. We assume that time is divided into discrete timeslots $\{1, 2, \ldots, L\}$. We are given a set of *demands* (alternatively, *jobs*) $\mathcal{D}$. Each demand $j \in \mathcal{D}$ has a starting timeslot $s(j)$, ending timeslot $e(j)$, bandwidth requirement $\rho(j)$ and profit $p(j)$. There are m *resources* (alternatively, *machines*) $\mathcal{P}_1, \ldots, \mathcal{P}_m$. Each resource $\mathcal{P}_i$ is described by a set of *resource-intervals*. Each resource-interval $I \in \mathcal{P}_i$ has a starting timeslot $s(I)$, ending timeslot $e(I)$, and offers a bandwidth $h(I)$. Let $\mathcal{P} = \cup_i \mathcal{P}_i$, (the set of all the resource-intervals – note that $\mathcal{P}$ could be a multiset).

A feasible solution $\mathcal{S}$ selects a subset of resource-intervals $R \subseteq \mathcal{P}$ and a subset of demands $J \subseteq \mathcal{D}$ satisfying the following constraints: (i) at most one resource-interval is selected for each resource $\mathcal{P}_i$; (ii) at any timeslot t, the sum of bandwidth requirements of jobs from J active at t is no more than the sum of the bandwidths offered by the resource-intervals of R active at t. The goal is to maximize the throughput; namely, sum of profits of jobs in J. We assume that starting times, ending times, bandwidth requirements of demands, bandwidths of resource-intervals are integers.

The RSTM problem is a generalization of the well-studied unsplittable flow problem (UFP) on line. In the UFP on line graphs, we are given a set of jobs; each job is specified by an interval, a bandwidth requirement and profit. For each timeslot, the input specifies the bandwidth available at the timeslot. The goal is to select a maximum profit subset of jobs such that the bandwidth constraints are not violated. The UFP is captured by the special case of the RSTM problem, wherein each resource consists of only a single resource-interval.

In addition to addressing the issue of selecting jobs as in the UFP, the RSTM problem also poses the challenge of scheduling the resources. However, the two tasks are interdependent, since the choice of resource scheduling critically determines the set of jobs that can be selected and hence, the profit of the solution. Thus, a key aspect of any algorithm for the RSTM problem lies in handling the two tasks simultaneously.

Our Results: We observe that the RSTM problem generalizes the maximum independent set problem, which is NP-hard to approximate within a factor of $n^{1-\epsilon}$, for any $\epsilon > 0$ [18]; see Appendix.

We study a restricted version of the RSTM problem, wherein the input instance satisfies the following two constraints: (i) the length of the longest job is at most the length of the shortest resource-interval (i.e., $\max_{j \in \mathcal{D}} \ell(j) \leq \min_{I \in \mathcal{P}} \ell(I)$, where $\ell(j) = e(j) - s(j) + 1$ and $\ell(I) = e(I) - s(I) + 1$); (ii) the bandwidth requirement of any job is at most the minimum bandwidth offered by the resource-intervals (i.e., $\max_{j \in \mathcal{D}} \rho(j) \leq \min_{I \in \mathcal{P}} h(I)$). Borrowing terminology from the UFP literature, we refer to the pair of constraints as the *no-bottleneck assumption* (NBA). Note that the second condition is analogous to the following condition required by known constant factor-approximation algorithms for UFP on the line – the maximum bandwidth requirement of any demand

is at most the smallest "edge capacity".

Our main result is a constant factor randomized approximation algorithm for the RSTM problem, under the NBA. We observe that dropping either one of the NBA constraints leads to the RSTM problem becoming as hard as the maximum independent set problem; see Appendix.

We now briefly discuss the main ideas behind our algorithm. A more detailed overview appears in Section 2. It can be shown that a natural LP relaxation for the problem has an unbounded integrality gap (see Appendix). Our constant factor approximation algorithm has two components: (i) we first show that any optimum solution can be transformed into a different solution having certain desirable structural properties (a so called "star solution"), with only a constant factor loss in profit; (ii) we next show that the optimum star solution can be approximated within constant factors in polynomial time – this is accomplished by applying randomized rounding to a suitable configuration LP.

Related Work: As discussed earlier, the RSTM problem is related to the UFP on a line. Calinescu et al. [5] and independently, Bar-Noy et al. [3] considered the uniform case of the UFP, where the bandwidth available is uniform across all time slots and gave a 3-approximation algorithm. The generalization where the bandwidth available can vary has received considerable attention. Constant factor approximation algorithms were known, under the NBA [9, 10]. Bansal et al. [2] presented a quasi-PTAS for the same problem, without assuming NBA; in a recent breakthrough, Bonsma et al. [4] designed a $(7+\epsilon)$-approximation algorithm for this problem.

Prior work has addressed an extension of the UFP on line with "bag constraints" on the demands. The bag constraint for a demand essentially specifies a set of intervals in which the demand can potentially be scheduled and the scheduler is allowed to pick at most one of these intervals. Bar-Noy et al. [3] presented a 5-approximation for this problem, when the bandwidth available is uniform across all timeslots. For the general case with arbitrary bandwidth availability, an $O(\log(\frac{h_{max}}{h_{min}}))$-approximation algorithm is known [8]; a constant factor approximation algorithm is known under the NBA assumption [7]. In our problem, the bag constraints are applied on the resources.

The model of limited machine availability, although rare, has been studied in the operations research literature. But most of these works do not consider the restriction of having to pick only one of the many intervals in which a machine is potentially available. Schmidt [16] considers the problem of scheduling jobs when there is a list of intervals of availability specified for each machine under the following objective functions: maximum completion time, sum of completion times, and maximum lateness. But the main difference with respect to our model is that, in Schmidt's model, a machine is available during all the intervals in its list. Aggoune [1] considers a model in which each machine has to shut down for certain maintenance tasks. The objective function considered is makespan minimization. Note that a machine is deemed available at all times other than its maintenance shut down. Motivated by applications in energy savings in data center operations, Khuller et al. [15] consider the following model. We are given a set of machines, where each machine has an activation cost and a limited budget is available for activating the machines. The goal is to activate the machines within the given budget and minimize the makespan of scheduling the jobs. They give a bi-criteria approximation algorithm for this problem. But, note that once a machine is activated, it is available at all the times.

2 Outline of Our Algorithm

We begin with a simplifying assumption that $\rho(j) = 1$ for all jobs j and $h(I)$ is an integer for all resource-intervals I. We will relax the assumption in Section 5.

Fix a feasible solution $\mathcal{S}$. We may also assume that for each job j (picked by $\mathcal{S}$) and each

timeslot t where j is active, the job is processed by a single resource-interval; we assume that the solution also specifies the resource-interval which processes j at each timeslot. We say that a resource-interval I *processes* a job j, if there is some timeslot t where I processes j.

As a warmup to our main result, let us first consider a simpler problem. The new problem, called single resource-interval `RSTM`, is the same as the `RSTM` problem, except that any job selected by the solution must be processed fully by a single resource-interval chosen by the solution. In contrast, a solution for the `RSTM` problem only needs to satisfy the bandwidth constraints, which means that a job may be jointly processed by multiple resource-intervals. It is easy to approximate the single resource-interval `RSTM` problem within a constant factor. The following theorem is proved in the Appendix.

Theorem 2.1 *The single resource-interval* `RSTM` *problem can be approximated in polynomial time within a factor of* 2*.*

We now turn our attention to the `RSTM` problem. As discussed earlier, the main issue is that a job may be processed by multiple resource-intervals. However, we will show that the optimum solution `opt` can be transformed into another solution with only a constant factor loss in profit such that any job in the transformed solution is processed by at most *two* resource-intervals; call such a solution a 2-*solution*. Similarly, a solution is said to be a 1-solution if each job is processed by a single resource-interval. For a 2-solution $\mathcal{X}$, let $J^{(1)}(\mathcal{X})$ denote the jobs processed by a single resource-interval and similarly, let $J^{(2)}(\mathcal{X})$ denote those processed by two resource-intervals. The set $J^{(2)}(\mathcal{X})$ can be represented as a graph wherein the resource-intervals selected by $\mathcal{X}$ form the vertices and an edge is drawn between two resource-intervals, if they jointly process some job (all such jobs can be labeled on the edge). We call this the *sharing graph* of $\mathcal{X}$ and denote it $G(\mathcal{X})$. We will show that with only a constant factor loss in profit, the solution $\mathcal{X}$ can be transformed further into two solutions $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$, where $\mathcal{S}_1^*$ is a 1-solution and $\mathcal{S}_2^*$ is a 2-solution whose sharing graph is a union of disjoint stars (a star consists of a *head* – or center – and a set of leaves connected to it). Furthermore, we show that in the solution $\mathcal{S}_2^*$, each resource-interval I will process at most $h(I)$ jobs from $J^{(2)}(\mathcal{S}_2^*)$. We call such a solution $\mathcal{S}_2^*$ as a *height-bounded star solution*. The theorem below summarizes the above transformations.

Theorem 2.2 *There exist a* 1*-solution* $\mathcal{S}_1^*$ *and a height-bounded star solution* $\mathcal{S}_2^*$ *such that* $\texttt{profit}(\mathcal{S}_1^*) + \texttt{profit}(J^{(2)}(\mathcal{S}_2^*)) \geq \texttt{profit}(\texttt{opt})/c$*, for a constant* c*.*

Theorem 2.1 implies the following corollary, which allows us to effectively approximate $\mathcal{S}_1^*$.

Corollary 2.3 *There exists a polynomial time algorithm that outputs a solution* $\mathcal{S}'$ *such that* $\texttt{profit}(\mathcal{S}') \geq \texttt{profit}(\mathcal{S})/2$*, for any* 1*-solution* $\mathcal{S}$*.*

Later, we shall design an algorithm for approximating the profit of $\mathcal{S}_2^*$ within constant factors and prove the following theorem (see Section 4).

Theorem 2.4 *There exists a polynomial time algorithm that outputs a solution* $\mathcal{S}'$ *such that* $\texttt{profit}(\mathcal{S}') \geq \texttt{profit}(J^{(2)}(\mathcal{S}))/c$*, for any height-bounded star solution* $\mathcal{S}$*, where* c *is some constant.*

The overall algorithm will simply output the best of the two solutions output by Corollary 2.3 and Theorem 2.4. It can be seen that the output solution $\mathcal{S}_{out}$ has profit at least a constant factor

of the profit of both $\mathcal{S}_1^*$ and $J^{(2)}(\mathcal{S}_2^*)$. This implies that $\texttt{profit}(\mathcal{S}_{out})$ is at least a constant factor of the profit of $\texttt{opt}$.

We now briefly outline the ideas behind the proof of Theorem 2.4. We write a *configuration LP* for finding the optimal height-bounded star solution. The LP essentially has exponentially many variables, one for each star. The separation oracle for this LP relies on finding the best *density* star solution – such a problem turns out to be a special case of submodular maximization subject to a matroid and two knapsack constraints [12]. Finally, we show that a simple randomized rounding algorithm with greedy updates gives a good integral solution.

3 Height-bounded Star Solutions: Proof of Theorem 2.2

We will first show that the optimum solution $\texttt{opt}$ can be transformed into a new 2-solution $\mathcal{S}$ with only a constant factor loss in profit. The proof exploits the first condition of the NBA (regarding the lengths of jobs). Furthermore, $\mathcal{S}$ will satisfy some additional properties.

3.1 Transformation to 2-solutions

Lemma 3.1 *There exists a 2-solution $\mathcal{S}$ having profit at least $\texttt{profit}(\texttt{opt})/4$. Furthermore, the solution $\mathcal{S}$ has the following properties:*

- *For any job $j \in J^{(2)}(\mathcal{S})$ processed by two resource-intervals I_1, I_2, where $s(I_1) \leq s(I_2)$, we have that $[s(j), e(j)]$ is not contained in either $[s(I_1), e(I_1)]$ and $[s(I_2), e(I_2)]$ (i.e., $s(I_1) < s(j) < s(I_2) \leq e(I_1) < e(j) < e(I_2)$); moreover, j is processed by I_1 till time $s(I_2)$ and by I_2 after this point of time.*
- *The total number of jobs from $J^{(2)}(\mathcal{S})$ processed by a resource-interval I is at most $2 \cdot h(I)$.*

Proof. We start by first proving a claim. Let X be a set of jobs. For a time t, let $A_t(X)$ denote the set of all jobs from X active at time t, i.e., the jobs j for which $s(j) \leq t \leq e(j)$.

Claim 3.2 *Given a set of jobs X, there exists a subset of jobs $Y \subseteq X$ such that for all time t, $|A_t(Y)| \leq 2$ and $|A_t(Y)| \geq \min\{1, |A_t(X)|\}$.*

Proof. We initialize a set Y to $\emptyset$. First select the job $j \in X$ with the smallest value of $s(j)$. In case there are more than one such jobs, choose one that has the largest value of $e(j)$. Add the selected job j to Y. We now repeatedly add jobs as follows. Let j denote the last job that was added to Y. Let $\mathcal{J}$ be the set of jobs j' for which $e(j') > e(j)$ and $s(j') \leq e(j)$. If $\mathcal{J}$ is not empty, add the job with the largest $e(j')$ value in $\mathcal{J}$ to Y. In case $\mathcal{J}$ is empty, consider the smallest time greater than $e(j)$ which is the start time of some job – let t denote this time. Let $\mathcal{J}'$ be the set of jobs which start at time t. Add the job in $\mathcal{J}'$ with largest end time to Y. This process is repeated as long as possible.

We now show that the set Y constructed above has the claimed property. Clearly, at every time t, $|A_t(Y)| \geq \min\{1, |A_t(X)|\}$. Now suppose that there are three jobs j_1, j_2 and j_3 (added in this order to Y) active at some time t. Consider the stage when j_3 was added to Y. As j_1 and j_2 were already present in Y, we have that $e(j_3) > e(j_2)$. Therefore, $s(j_3) \leq e(j_1) < e(j_3)$. Now consider the stage when j_2 was added. Notice that j_3 overlaps with j_1 (and hence, also with the last job added to Y). Moreover, j_3 extends further than j_2. Hence, j_3 should have been added instead of j_2. This proves the claim. ∎

We now complete the proof of the lemma. For each resource-interval I selected by `opt`, we replace it by $h(I)$ new resource-intervals of unit capacity each – the start and end times of these new resource-intervals are same as those of I respectively. We call these resource-intervals *slices* of I. Let $\mathcal{B}$ denote the set of all slices obtained in this way. For a subset of slices $L \subseteq \mathcal{B}$, let $A_t(L)$ denote the set of slices active at time t. Using an argument similar to that of Claim 3.2, we can find a subset of slices $L \subseteq \mathcal{B}$ such that at each time $|A_t(L)| \leq 2$ and $|A_t(L)| \geq \min\{1, |A_t(\mathcal{B})|\}$. This is achieved using the same procedure as in the proof of Claim 3.2.

Let J be the set of all jobs selected by `opt`. We next apply Claim 3.2 twice: apply the claim on the set J to get a set of jobs Y_1 and then apply the claim again on the set $J - Y_1$ to get a set of jobs Y_2. Let $Y = Y_1 \cup Y_2$. The set Y satisfies the property that at each time $|A_t(Y)| \leq 4$ and $|A_t(Y)| \geq \min\{2, |A_t(J)|\}$.

At any time t, at most four jobs from Y are active. This means that the set Y can be colored with four colors, i.e., Y can be partitioned into four subsets such that no two jobs from the same subset are active at the same time. Let Y' be the subset having the maximum profit among these four subsets. Then, $\texttt{profit}(Y') \geq \texttt{profit}(Y)/4$. We now assign jobs in Y' to the slices in L. Consider a job $j \in Y'$. Since this job is part of the feasible solution $\mathcal{S}$, $A_t(\mathcal{B}) \geq 1$ for all $t \in [s_j, e_j]$. Therefore, for all such time t, $|A_t(L)| \geq 1$. Among the slices $\ell \in L$ active at $s(j)$, pick the slice having the largest $e(\ell)$. If ℓ fully contains j, then simply assign j to ℓ. Otherwise (i.e., if $e(j) > e(\ell)$), there must be some slice $\ell' \in L$ which is active at time $e(\ell)$ and $e(\ell') > e(\ell)$. Note that $s(\ell') > s(j)$ because of our choice of ℓ. Moreover, $e(\ell') > e(j)$ because of the no bottleneck assumption. Thus, j can be assigned to ℓ and ℓ'. In this case, we say that ℓ is the *left interval* used by j and ℓ' is the *right interval* used by j. In this way, we are able to assign every job in Y' to at most two slices (and hence, the corresponding resource-intervals) in L.

Now, delete all the jobs in Y from J and all the slices in L from $\mathcal{B}$ – let J' and $\mathcal{B}'$ denote these new sets respectively. We note that the jobs in J' "fit" in the slices of $\mathcal{B}'$, i.e., at any time t, $|A_t(J')| \leq |A_t(\mathcal{B}')|$. This follows from the fact that at any time t, $|A_t(Y)| \geq \min\{|A_t(L)|, |A_t(J)|\}$. We can now apply the entire procedure iteratively with these new sets J' and $\mathcal{B}'$. We collect all the sets Y' generated in each of these iterations and take Z to be the union of these sets. We see that $\texttt{profit}(Z) \geq \texttt{profit}(J)/4$ and each job $j \in Z$ is assigned to at most two resources.

Each resource-interval I is used as a left interval by at most $h(I)$ jobs and as a right interval by at most $h(I)$ jobs. Note that all the properties stated in the theorem are satisfied. ∎

Let $\mathcal{S}$ be the solution guaranteed by Lemma 3.1. The lemma implies that the jobs serviced by the solution $\mathcal{S}$ can only be of two types – $J^{(1)}(\mathcal{S})$, which are processed by exactly one resource-interval in $\mathcal{S}$; and $J^{(2)}(\mathcal{S})$, which are serviced by two resource-intervals selected in $\mathcal{S}$. Let $\mathcal{C}$ be the resource-intervals selected by $\mathcal{S}$. A job $j \in J^{(2)}(\mathcal{S})$ uses two resource-intervals $I_1, I_2 \in \mathcal{C}$ as stated in Lemma 3.1 – let I_1 be the *left interval* used by j and I_2 be the *right interval* used by j. We also use the terminology that I_1 (or I_2) services j as its left (or right) interval. We call a solution $\mathcal{X}$ to be *nice*, if the following properties hold: (i) $\mathcal{X}$ satisfies the conditions of Lemma 3.1; (ii) every resource-interval I selected by $\mathcal{X}$ services all jobs in $J^{(2)}(\mathcal{X})$ as either their left interval or their right interval only; (iii) the number of jobs processed by a resource-interval I from $J^{(2)}(\mathcal{X})$ is at most $h(I)$.

Lemma 3.3 *Let $\mathcal{S}$ be a solution as guaranteed by Lemma 3.1. Then there exists a* nice *solution $\mathcal{S}'$ such that* $\texttt{profit}(\mathcal{S}') \geq \texttt{profit}(J^{(1)}(\mathcal{S})) + \texttt{profit}(J^{(2)}(\mathcal{S}))/4$.

Proof. We independently label each resource-interval I selected by $\mathcal{S}$ as either $\mathcal{L}$ or $\mathcal{R}$ with probability half. We process a job $j \in J^{(2)}(\mathcal{S})$ if and only if its left interval gets label $\mathcal{L}$ and

its right interval gets label $\mathcal{R}$. Clearly, such a solution has the desired properties. The second statement in the lemma follows from the proof of Lemma 3.1. Each resource-interval I in $\mathcal{S}$ acts as left interval (or right interval) for at most $h(I)$ jobs. The jobs in $J^{(1)}(\mathcal{S})$ are retained in $\mathcal{S}'$. Now, note that a job $j \in J^{(2)}(\mathcal{S})$ gets selected in $\mathcal{S}'$ with probability $\frac{1}{4}$. Thus, the expected profit of $\mathcal{S}'$ is at least $\texttt{profit}(J^{(1)}(\mathcal{S})) + \texttt{profit}(J^{(2)}(\mathcal{S}))/4$. This shows the existence of the claimed solution. ∎

3.2 Transformation to Height-Bounded Star Solutions

By Lemma 3.3, we can assume the existence of a nice solution $\mathcal{S}$ having profit at least a constant factor of the optimum solution. The sharing graph $G(\mathcal{S})$ is a bipartite graph with the two sides being denoted $\mathcal{L}(S)$ and $\mathcal{R}(\mathcal{S})$ – the resource-intervals which serve jobs as their left interval and those which serve jobs as their right interval respectively. Our next result shows that the sharing graph $G(\mathcal{S})$ can be assumed to be a star (with only a constant factor loss in profit).

Lemma 3.4 *Given a nice solution $\mathcal{S}$, there is another solution $\mathcal{S}' \subseteq J^{(2)}(\mathcal{S})$ such that $\mathcal{S}'$ is a height-bounded star solution and the profit of $\mathcal{S}'$ is at least half of the profit of $J^{(2)}(\mathcal{S})$.*

Proof. Let $\mathcal{C}$ be the resource-intervals selected by $\mathcal{S}$. We first reduce the capacity $h(I)$ of a resource-interval in $\mathcal{C}$ to the number of jobs in $J^{(2)}(\mathcal{S})$ that it services (Lemma 3.3 shows that this quantity is at most $h(I)$). Also note that this is just for the sake of studying the property of the solution $\mathcal{S}$. We now prove that it is possible to process the jobs in $J^{(2)}(\mathcal{S})$ in $\mathcal{C}$ such that the sharing graph is a forest. To this end, we explicitly give a way of packing these jobs in the resource-intervals in $\mathcal{S}$.

We arrange the jobs $j_1, \ldots, j_n$ in ascending order of $s(j)$, i.e., $s(j_1) \leq s(j_2) \leq \cdots \leq s(j_n)$. We arrange the resource-intervals in $\mathcal{L}(\mathcal{S})$ in ascending order of $e(I)$ values, i.e., $e(I_1^L) \leq e(I_2^L) \cdots \leq e(I_l^L)$, where $l = |\mathcal{L}(\mathcal{S})|$ and $\mathcal{L}(\mathcal{S}) = \{I_1^L, \ldots, I_l^L\}$. Similarly, we arrange the resource-intervals in $\mathcal{R}(\mathcal{S}) = \{I_1^R, \ldots, I_r^R\}, r = |\mathcal{R}(\mathcal{S})|$ in ascending order of $s(I)$ values. For an integer c, $1 \leq c \leq n$, let $L(c)$ be the unique integer i satisfying $h(I_1^L) + \cdots + h(I_{i-1}^L) < c \leq h(I_1^L) + \cdots + h(I_{i-1}^L) + h(I_i^L)$. Define $R(c)$ similarly for the resource-intervals in $\mathcal{R}(\mathcal{S})$.

Claim 3.5 *For $1 \leq c \leq n$, the job j_c can be processed by the pair of resource-intervals $(I_{L(c)}^L, I_{R(c)}^R)$.*

Proof. Let a denote $L(c)$ and b denote $R(c)$. We need to prove that j_c is contained in the span of I_a^L and I_b^R; this means we have to prove: (i) $s(j_c) \geq s(I_a^L)$; (ii) $e(j_c) \leq e(I_b^R)$; (iii) $s(I_b^R) \leq e(I_a^L) + 1$ (i.e., there should not exist a timeslot – a gap – between the ending timeslot of I_a^L and the starting timeslot of I_b^R). We prove these below:

- Suppose, for the sake of contradiction, $s(j_c) < s(I_a^L)$. Then for any $c' \leq c$, $s(j_{c'}) < s(I_a^L)$. The no bottleneck assumption implies that $e(j_{c'}) < e(I_a^L)$ and so, $e(j_{c'}) < e(I_{a'}^L)$ for all $a' \geq a$. But then $\mathcal{S}$ must have scheduled $j_{c'}$ using one of the resource-intervals in $\{I_1^L, \ldots, I_{a-1}^L\}$. But this is not possible because there are c jobs in $\{j_1, \ldots, j_c\}$ and the total capacity of $\{I_1^L, \ldots, I_{a-1}^L\}$ is less than c. This shows that $s(j_c) \geq s(I_a^L)$.

- Again, suppose $e(j_c) > e(I_b^R)$. Then, the no bottleneck assumption implies that $s(j_c) > s(I_b^R)$ and so, $s(j_{c'}) > s(I_{b'}^R)$, if $c \leq c'$ and $b' \leq b$. So the jobs in $\{j_c, \ldots, j_n\}$ can only be scheduled among $I_{b+1}^R, \ldots, I_r^R$. Now, the fact $h(I_1^R) + h(I_2^R) + \cdots + h(I_r^R) = n$ and the definition of b lead to a contradiction.

- Since $s(j_c) < e(j_c)$, the above two statements imply that I_a^L and I_b^R have a non-trivial intersection iff $e(I_a^L) > s(I_b^R)$. If $e(I_a^L) < s(I_b^R)$, then $e(I_{a'}^L) < s(I_{b'}^R)$, $a' \leq a, b' \geq b$. So, the

resource-intervals in $\{I_1^L, \ldots, I_a^L\}$ can have non-trivial intersection with resource-intervals in $\{I_1^R, \ldots, I_{b-1}^R\}$ only. But the former set of resource-intervals are processing at least c jobs and the latter set can process less than c jobs – a contradiction.

∎

Consider the schedule implied by the claim above. Note that the sharing graph is bipartite. Also, it is easy to check that in the sharing graph, one of the vertices I_1^L and I_1^R is a leaf. Removing this leaf and continuing the argument gives a sequence in which we can remove leaves from the sharing graph (till all vertices are removed). So it must be a forest. Now consider a tree in the forest. We can decompose the edges into two sets of disjoint union of stars (just consider alternate levels). Then, we pick the set with the higher profit. The jobs in $J^{(1)}(\mathcal{S})$ are retained in $\mathcal{S}'$. ∎

Proof of Theorem 2.2: Let $\mathcal{S}$ be the nice solution output by Lemma 3.3. We take $\mathcal{S}_1^* = J^{(1)}(\mathcal{S})$ and take the solution $\mathcal{S}'$ output by Lemma 3.4 as $\mathcal{S}_2^*$.

4 Finding Height-bounded Star Solutions: Proof of Theorem 2.4

For a star T, let $\texttt{hd}(T)$ be the root vertex in T, and $\texttt{Leaves}(T)$ be the set of leaves in T. We say that a star T can *satisfy* a set of jobs L if it is possible to partition L into $|\texttt{Leaves}(T)|$ sets – $L_1, \ldots, L_r$, $r = |\texttt{Leaves}(T)|$, such that the following conditions holds :

- $|L| \leq h(\texttt{hd}(T))$.
- Let T_i be the i^{th} leaf of T. Then for $i = 1, \ldots, |\texttt{Leaves}(T)|$, $|L_i| \leq h(T_i)$ and each job in L_i can be served jointly by $\texttt{hd}(T)$ and T_i.

We say that a collection $\{(T^{(1)}, L^{(1)}), (T^{(2)}, L^{(2)}), \ldots, (T^{(t)}, L^{(t)})\}$, is a *satisfiable star solution*, if $T^{(i)}$ can satisfy $L^{(i)}$, for all i. The main result of this section shows that the optimum satisfiable star solution can be approximated within constant factors.

Theorem 4.1 *There exists a randomized polynomial time algorithm that outputs a satisfiable star solution whose profit is at least a constant factor of the optimum satisfiable star solution.*

It is clear any satisfiable star solution is a feasible height-bounded star solution. Furthermore, for any height-bounded star solution $\mathcal{S}$, $J^{(2)}(\mathcal{S})$ is a satisfiable star solution. Therefore, the above result implies Theorem 2.4. The rest of the section is devoted to proving Theorem 4.1. The algorithm is based on LP rounding.

4.1 LP Relaxation

We now write an LP relaxation for finding the optimal satisfiable star solution. For each pair (T, L), where T is a star and L is a set of jobs which can be satisfied by T, we have a variable $y(T, L)$. For a set of jobs L, define $p(L)$ as the total profit of jobs in L. The LP relaxation is as follows :

$$\begin{aligned}
\max \quad & \sum_{(T,L)} y(T,L) \cdot p(L) & & \\
& \sum_{(T,L): j \in L} y(T,L) \leq 1 & \text{for all } j \in \mathcal{D} & \quad (1) \\
& \sum_{(T,L): T \cap \mathcal{P}_i \neq \emptyset} y(T,L) \leq 1 & \text{for all } \mathcal{P}_i & \quad (2) \\
& y(T,L) \geq 0 & \text{for all } (T,L) &
\end{aligned}$$

Now, there are two issues – solving the LP (since the number of variables is exponential) and rounding a fractional solution. We approximately solve the LP by providing a approximate separation oracle for the dual. The oracle goes by framing the issue as maximizing a submodular function subject to a matroid constraint and two knapsack constraints. Recent results [13] provide approximation algorithms for the latter task. The details are in the Appendix. We now proceed to show how to round the fractional solution.

4.2 Rounding a fractional solution

Let $y^*(T, L)$ be an optimal (or near-optimal) solution to the LP. We show how it can be rounded to an integral solution. We can assume that $y^*(T, L)$ will assume non-zero values for a polynomial number of pairs (T, L) only – let these be $(T^{(r)}, L^{(r)}), r = 1, \ldots, n$. Let the leaves of $T^{(r)}$ be labeled $I_1^{(r)}, \ldots, I_k^{(r)}$, where $k = |\texttt{Leaves}(T^{(r)})|$. Since $T^{(r)}$ can satisfy $L^{(r)}$, we divide these jobs into $|\texttt{Leaves}(T^{(r)})|$ sets – $L_u^{(r)}$, $u = 1, \ldots, L_k^{(r)}$, such that $L_u^{(r)}$ can be processed by $\texttt{hd}(T^{(r)})$ and the u^{th} leaf of this star. Consider the algorithm given in Figure 1.

For $r = 1, \ldots, n$ do

With probability $\frac{y^*(T^{(r)}, L^{(r)})}{12}$, consider the pair $(T^{(r)}, L^{(r)})$ in the following steps.

1. If no resource-interval from the resource containing $\texttt{hd}(T^{(r)})$ has been chosen far
   (i) Select the resource-interval $\texttt{hd}(T^{(r)})$.
   (ii) For $i = 1, \ldots, |\texttt{Leaves}(T^{(r)}|$ do
   If no resource-interval from the resource containing $I_i^{(r)}$ has been taken yet, then select the resource-interval $I_i^{(r)}$ and service the jobs in $L_i^{(r)}$ using this resource-interval and $\texttt{hd}(T^{(r)})$.

Figure 1: Algorithm `Round`

We now argue that the expected profit of this solution is at least a constant times that of the fractional solution y^*. For a job j, define Δ_j as $\sum_{(T,L):j \in L} y^*(T, L)$. It is easy to check that

$$\sum_{(T,L)} y^*(T, L) \cdot p(L) = \sum_j \Delta_j \cdot p(j).$$

Thus it is enough to prove that the probability we service j is $\Omega(\Delta_j)$. We fix a job j. Assume without loss of generality that j appears in $L^{(1)}, \ldots, L^{(t)}$. Further, assume that j appears in the subset $I_1^{(l)}$ of $L^{(l)}$, $1 \le l \le t$. For sake of brevity, let z_l denote $\frac{y^*(T^{(l)}, L^{(l)})}{12}$, and z denote $\sum_l z_l$. Let X_l, $1 \le l \le t$, denote the 0-1 random variable which is 1 precisely when we select both $\texttt{hd}(L^{(l)})$ and the leaf interval $I_1^{(l)}$. Thus, we service j iff at least one of the random variables X_l is 1.

Claim 4.2 *For $1 \le l \le t$, we have $z_l/2 \le \mathbf{E}[X_l] \le z_l$.*

Proof. First note that $X_l = 1$ only if we choose to consider the pair $(T^{(l)}, L^{(l)})$. So $E[X_l] \le z_l$. Now suppose we consider this pair (which happens with probability z_l). Constraint (2) implies that the probability that we do not select $\texttt{hd}(L^{(l)})$ is at most $\frac{1}{12}$, and the same statement holds for $I_1^{(l)}$. So, the probability that we do not select one of these two resource-intervals is at most $\frac{1}{6}$. Therefore, X_l is equal to 1 with probability at least $\frac{5}{6} \cdot z_l$. ∎

Now we would like to bound the probability that all X_l are 0. Unfortunately, the random variables X_l are not independent. Define $X = \sum_l X_l$. The claim above shows that $E[X] \geq \frac{z}{2}$.

Claim 4.3 *For all integers $p \geq 1$, $\mathbf{Pr}[X = p] \leq z^p$.*

Proof. Fix any p random variables $X_{i_1}, \ldots, X_{i_p}$. The probability that all of these are 1 is at most $\Pi_{l=1}^{p} z_{i_l}$ (since the coin tosses for deciding whether we consider a pair (T, l) are independent). Now taking the sum over all such tuples, we see that $\mathbf{Pr}[X = p]$ is at most $(z_1 + \cdots + z_t)^p$. ∎

We are now almost done.

$$\begin{aligned} \frac{z}{2} \leq \mathbf{E}[X] &= \sum_{p=1}^{\infty} p \cdot \mathbf{Pr}[X = p] \leq \mathbf{Pr}[X = 1] + \sum_{p \geq 2} p \cdot z^p \\ &\leq \mathbf{Pr}[X = 1] + 3z^2 \leq \mathbf{Pr}[X = 1] + \frac{z}{4} \end{aligned}$$

where the last inequality is true since $z = \frac{\Delta_j}{12} \leq \frac{1}{12}$ (using constraint (1)). Thus we get $\mathbf{Pr}[X = 1] \geq \frac{z}{4} = \Omega(\Delta_j)$, which is what we wanted to prove. So, the expected profit of algorithm `Round` is at least a constant times that of y^*. This completes the proof of Theorem 4.1.

5 Overall Algorithm

We finally put everything together to get the following main result.

Theorem 5.1 *There exists a constant factor randomized approximation algorithm for the* `RSTM` *problem (with* NBA*).*

Proof. The overall algorithm (called `Schedule`) outputs the best of the two solutions output by Corollary 2.3 and Theorem 2.4. Let the solution output by the our algorithm be $\mathcal{S}$ and let `opt` be the optimum solution. By Theorem 2.2, `opt` can be transformed into a height-bounded star solution $\mathcal{S}^*$, with only a constant factor loss in profit. It can be seen that the output solution $\mathcal{S}$ has profit at least a constant factor of the profit of both $J^{(1)}(\mathcal{S}^*)$ and $J^{(2)}(\mathcal{S}^*)$. This implies that `profit`$(\mathcal{S})$ is at least a constant factor of the profit of $\mathcal{S}^*$. Hence, $\mathcal{S}$ is a constant factor approximation to `opt`.

Now consider the general case when jobs can have arbitrary (but integral) $\rho(j)$ values. We will assume that $h(I)$ is an integer for all resource-intervals I. We divide a job j into $\rho(j)$ new jobs – each such job j' has $\rho(j') = 1$ and profit equal to $\frac{p(j)}{\rho(j)}$. We now run the algorithm `Schedule`on this instance. Now, the problem is that a job j may get picked to a *partial* extent, i.e., the algorithm may pick $r(j)$ copies of the jobs, where $0 \leq r(j) \leq \rho(j)$, and the corresponding profit is $\frac{r(j)}{\rho(j)} \cdot p(j)$. Now, consider the resource-intervals picked by this algorithm – this gives a certain capacity to each timeslot (which is equal to the capacities of resource-intervals which get picked and contain this timeslot). So the solution can be thought of as a *fractional* solution to unsplittable-flow problem (`UFP`) on the line where the capacities of timeslots are as described, and a job j is picked to an extent of $\frac{s(j)}{\rho(j)}$. But we know that the standard LP relaxation for `UFP` has constant integrality gap [10]; so, we can find an integral solution (i.e., either we pick the entire job or none of it) of profit at least a constant times that of the solution output by algorithm `Schedule`. ∎

References

[1] R. Aggoune. Minimizing the makespan for the flow shop scheduling problem with availability constraints. *European Journal of Operations Research*, 125:534–543, 2004.

[2] N. Bansal, A. Chakrabarti, A. Epstein, and B. Schieber. A quasi-ptas for unsplittable flow on line graphs. In *ACM Symposium on Theory of Computing*, pages 721–729, 2006.

[3] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Noar, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, 2001.

[4] P. Bonsma, J. Schulz, and A. Wiese. A constant factor approximation algorithm for unsplittable flow on paths. *CoRR*, abs/1102.3643, 2011.

[5] G. Calinescu, A. Chakrabarti, H. Karloff, and Y. Rabani. Improved approximation algorithms for resource allocation. In *Prooceedings of the 9th International Conference on Interger Programming and Combinatorial Optimization*, 2002.

[6] M. Carlisle and E. Lloyd. On the k-coloring of intervals. *Discrete Applied Mathematics*, 59(3):225–235, 1995.

[7] V. Chakaravarthy, A. Choudhury, and Y. Sabharwal. A near-linear time constant factor algorithm for unsplittable flow problem on line with bag constraints. In *FSTTCS*, 2010.

[8] V. Chakaravarthy, V. Pandit, Y. Sabharwal, and D. Seetharam. Varying bandwidth resource allocation problem with bag constraints. In *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing*, 2010.

[9] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation algorithms for the unsplittable flow problem. *Algorithmica*, 47(1):53–78, 2007.

[10] C. Chekuri, M. Mydlarz, and F. Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Transactions on Algorithms*, 3(3), 2007.

[11] Chandra Chekuri and Amit Kumar. Maximum coverage problem with group budget constraints and applications. In *APPROX-RANDOM*, pages 72–83, 2004.

[12] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS*, 2010.

[13] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *WINE*, 2010.

[14] N. Hatziargyriou, H. Asano, R. Iravani, and C. Marnay. Microgrids. *IEEE Power and Energy Magazine*, 5(4):78–94, 2007.

[15] S. Khuller, J. Li, and B. Saha. Energy efficient scheduling via partial shutdown. In *Proceedings of the ACM-SIAM Symosium on Discrete Algorithms (SODA)*, 2010.

[16] G. Schmidt. Scheduling with limited machine availability. *European Journal of Operations Research*, 121:1–15, 2000.

[17] M. Yannakakis and F. Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24(2):133–137, 1987.

[18] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *STOC*, 2006.

A Hardness Results

Our main result presents a constant factor approximation algorithm for the RSTM problem with NBA. Here, we show that the RSTM problem without NBA is as hard to approximate as the maximum independent set problem. We also show that the RSTM problem along with any one of the two NBA constraints is also equally hard. This shows that both the NBA constraints are needed to get a constant factor approximation algorithm.

Theorem A.1 *The* RSTM *problem is NP-hard to approximate within a factor of* $n^{1-\epsilon}$*, for any* $\epsilon > 0$ *(where* n *is the number of jobs in the input).*

Proof. We give a reduction from the maximum independent set problem to the RSTM problem. Let $G = (V, E)$ be the input graph. Arrange the vertices of G in some arbitrary order $u_1, u_2, \ldots, u_n$. Let $\deg(u_i)$ denote the degree of the vertex u_i. Let the $c_i = \sum_{1 \leq j \leq i-1} \deg(u_j)$ denote the sum of degrees of vertices preceding u_i in the above ordering. Construct an RSTM problem instance as follows. For each vertex u_i, add a job u_i which is set to be active in the timeslots $\{c_i+1, c_i+2, \ldots, c_i+\deg(u_i)\}$. The bandwidth requirements of all jobs is 1. Due to the correspondence between vertices and jobs, we shall view vertices as jobs, and vice versa. For each edge $e = (u, v)$ in G add a resource $\mathcal{P}_e$ of length 1 and offering a bandwidth 1; the resource can be scheduled in any one of the timeslots where u or v is active. This completes the construction. Notice that if (u, v) is an edge in G, then any feasible solution can select at most one of the two jobs u and v. Meaning, if J is a feasible set of jobs, then J must be an independent set in G. Conversely, if J is an independent in G, then we can construct a feasible solution in which all the jobs in J are selected. It follows that G has an independent set of size k, if and only if there exists a feasible solution of profit k. The theorem follows the known hardness result for the maximum independent set problem [18]. ∎

The RSTM problem instances constructed by the above reduction satisfies the second constraint of the NBA (though the first constraint of NBA is violated). Thus, the same proof establishes the following result.

Theorem A.2 *Consider the restricted version of the* RSTM *problem wherein the input instances satisfy the second constraint of the* NBA*. This restricted problem is also NP-hard to approximate within a factor of* $n^{1-\epsilon}$*, for any* $\epsilon > 0$ *(where* n *is the number of jobs in the input).*

We shall now provide a similar construction to show that the RSTM problem remains hard, even if input instances satisfy the first constraint of NBA.

Theorem A.3 *Consider the restricted version of the* RSTM *problem wherein the input instances satisfy the first constraint of the* NBA*. This restricted problem is also NP-hard to approximate within a factor of* $n^{1-\epsilon}$*, for any* $\epsilon > 0$ *(where* n *is the number of jobs in the input).*

Proof. We give a reduction from the maximum independent set problem. Let $G = (V, E)$ be the input graph. Arrange the vertices in an arbitrary order $u_1, u_2, \ldots, u_n$. Let $\deg(u_i)$ denote the

degree of the vertex u_i. Construct an RSTM problem instance as follows. For each vertex u_i, add a job u_i which is set to be active in single timeslot $\{i\}$. The bandwidth requirements of the job u_i is set as $\deg(u_i)$. Due to the correspondence between vertices and jobs, we shall view vertices as jobs, and vice versa. For each edge $e = (u, v)$ in G add a resource $\mathcal{P}_e$ of length 1 and offering a bandwidth 1; the resource can be scheduled in any one of the two timeslots where u or v is active. This completes the construction. Notice that if (u, v) is an edge in G, then any feasible solution can select at most one of the two jobs u and v. Meaning, if J is a feasible set of jobs, then J must be an independent set in G. Conversely, if J is an independent in G, then we can construct a feasible solution in which all the jobs in J are selected. It follows that G has an independent set of size k, if and only if there exists a feasible solution of profit k. The theorem follows the known hardness result for the maximum independent set problem [18]. ∎

B Integrality Gap for a Natural LP Relaxation

We first consider the natural LP relaxation for the problem and show that it has unbounded integrality gap. In this LP formulation, we associate a variable $y(I)$ with each resource-interval $I \in \mathcal{P}$ and a variable $x(j)$ with each job $j \in \mathcal{D}$. Further, for each job j, each resource-interval I and each timeslot t such that I and j are active at t, we associate a variable $x(I, j, t)$; this signifies the fraction of the job j that is allocated to resource-interval I at timeslot t. The LP relaxation is as below.

$$
\begin{aligned}
\max \quad & \sum_{j} x(j) \cdot p(j) & & & \\
& \sum_{I} x(I,j,t) & = & x(j) \leq 1 & \text{for all } j \in \mathcal{D}, \text{ for all timeslots } t \\
\sum_{j \,:\, s(j) \leq t \leq e(j)} & x(I,j,t)\rho(j) & \leq & y(I)h(I) & \text{for all } I \in \mathcal{P}, \text{ for all timeslots } t \\
& 0 \leq x(I,j,t) & \leq & y(I) & \text{for all } I,\ j,\ t \\
& \sum_{I \in \mathcal{P}_i} y(I) & \leq & 1 & \text{for all resources } \mathcal{P}_i
\end{aligned}
$$

We now show that the LP has unbounded integrality gap. Fix any parameter B. Consider an input instance consisting of two resources $\mathcal{P}_1$ and $\mathcal{P}_2$, and B jobs $j_1, j_2, \ldots, j_B$. The resource $\mathcal{P}_1$ contains two resource-intervals I_1 and I_2, both offering bandwidth B. I_1 is active at timeslots $\{1, 2, 3, 4\}$ and I_2 is active at timeslots $\{5, 6, 7, 8\}$. The resource $\mathcal{P}_2$ has a single resource-interval I_3, offering bandwidth 1 and active at timeslots $\{5, 6, 7, 8\}$. All jobs are active at timeslots $\{3, 4, 5, 6\}$, have profit 1 and bandwidth requirement of 1 each. Notice that this input instance satisfies both the constraints of the NBA. Clearly, the integral optimal solution can select only one job and so, it has profit 1. On the other hand, the LP relaxation can set $y(I_1) = y(I_2) = 1/2$ and for all jobs j, set $x(j) = 1/2$. The other variables can be set suitably so as to satisfy all the linear constraints. Thus, the LP solution has a profit of $B/2$. The fundamental issue with the above LP is that it allows a job to be assigned to multiple resource-intervals of the same resource.

One can imagine writing another "natural LP" relaxation which relies on only the fact that a job is assigned to at most two resource-intervals – it will have variables $x(j, I, I')$ which denotes the fact that x is assigned to I and I' only (and so I, I' are from different resources). It is not difficult

to give an integrality gap example. The principal reason for this is that a resource-interval I can appear in several pairs, whereas in the configuration LP, a resource-interval can appear in only *one* star.

C Solving the LP

Here, we show that there is a polynomial time (approximate) separation oracle for the dual LP. We have dual variables α_j corresponding to constraint (1) and β_i for constraint (2). The dual LP is

$$\begin{aligned} \min \quad & \sum_j \alpha_j + \sum_i \beta_i \\ \sum_{i:T\cap\mathcal{P}_i\neq\emptyset} \beta_i + \sum_{j\in L} \alpha_j \quad & \geq \quad p(L) \qquad \text{for all } (T,L) \qquad (3) \\ \alpha_j, \beta_i \quad & \geq \quad 0 \end{aligned}$$

Suppose we are given a solution α_i, β_j to the dual LP. Let $\mathcal{D}'$ be the set of jobs for which $\alpha_j \leq p(j)$. We need to check if there is a pair (T, L) such that constraint (3) corresponding to this pair is violated. Suppose, for the sake of argument, that there is such a violating pair. We first argue that $L \subseteq \mathcal{D}'$. Indeed, if $j \in L$ and $p(j) < \alpha_j$, then the pair $(T, L - \{j\})$ will also be a violating pair, and so we can replace L by $L - \{j\}$. Continuing this argument, we can assume that $L \subseteq \mathcal{D}'$. For a job $j \in \mathcal{D}'$, define $p'(j)$ as $p(j) - \alpha_j$. For a subset L of $\mathcal{D}'$, define $p'(L)$ analogously. For a resource-interval $I \in \mathcal{P}_i$, extend the definition β_i to I as $\beta(I) = \beta_i$. For a star T, define $\beta(T) = \sum_{I\in T} \beta(I)$. So now, finding a violating set is subsumed by the problem of finding a pair (T, L), $L \subseteq \mathcal{D}'$, such that the *density* $\frac{\beta(T)}{p'(L)}$ is minimized. We now prove that this problem can be solved approximately. Suppose (T^*, L^*) achieves the desired minimum. We guess the resource-interval $\texttt{hd}(T^*)$ and the quantity $\beta(T^*)$ up to a factor of 2 – call this Q. Clearly, it is enough to prove the following lemma. The proof goes by framing the problem as that of maximizing a submodular function subject to a matroid constraint and two knapsack constraints (see Appendix for a proof).

Lemma C.1 *Given a resource-interval N and quantity Q, there is a constant factor approximation algorithm for finding a star T and set of jobs $L \subseteq \mathcal{D}'$ where T can satisfy L, $\texttt{hd}(T) = N$, $\beta(T) \leq Q$, such that $p'(L)$ is maximized.*

Proof. Let $\mathcal{P}_i$ be the resource containing N. We frame the problem as that of maximizing a submodular function subject to a matroid constraint and two knapsack constraints. The desired result then follows from [12, 13]. We create a bipartite graph on a vertex set $A \cup B$ as follows. A is the set of jobs in $\mathcal{D}'$. For each $I, I \notin \mathcal{P}_i$, we have $h(I)$ vertices in B, which we label as $v(I, r), r = 1, \ldots, h(I)$. Intuitively, we will use $v(I, r)$ if I is assigned *exactly* r jobs. For a job $j \in A$ and resource-interval $I \in B$, we have an edge from j to $v(I, r)$ for all r if j can be scheduled by N and I, i.e., the *span* of j is contained in the union of that of N and I. For a subset S of B, let $f(S)$ denote the maximum value of $p'(D)$, $D \subseteq A, |D| \leq h(N)$, such that there is a b-matching between S and D which satisfies that the degree of $v(I, r) \in S$ in this b-matching is at most r and that of a job in D is 1. It is not difficult to show that f is a submodular function. Thus the desired problem is to find a a set $S \subseteq B$ such that $f(S)$ is maximized and (i) $|S \cap \{v(I, r) : I \in \mathcal{P}_l, 1 \leq r \leq h(I)\}| \leq 1$ for all $l \neq i$, (ii) $\sum_{v(I,r)\in S} r \leq h(N)$, and (iii) $\beta(N) + \beta(S) \leq Q$. Here $\beta(v(I, r))$ is defined to be same as $\beta(I)$. The first constraint is a partition matroid constraint and the other two are knapsack

constraints. Therefore, we can get a constant factor approximation algorithm for this problem [12, 13]. This completes the proof. ∎

The lemma above shows that we can get a separation oracle (upto constant factor) for the dual LP. Thus we can get a solution to the LP whose value is at least a constant times the optimal value.

D Proof of Theorem 2.1

For a fixed resource-interval I, the problem of finding the largest profit set of jobs which can be satisfied by only this resource-interval is equivalent to the problem of finding the maximum weight k-colorable subgraphs in interval graphs, with $k = h(I)$ (because of our assumption that $\rho(j) = 1$ for all jobs j). The latter problem can be solved optimally in polynomial time [17, 6]. Now consider the following greedy algorithm – consider the resources in some order, say $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m$. When considering $\mathcal{P}_i$, select the resource-interval which can schedule the maximum profit set of remaining jobs (here, we use the optimal algorithm mentioned above). Remove the jobs which get scheduled in this resource-interval. It can be shown that the profit of this solution is at least half the profit of the optimum – indeed this falls under the framework of maximum coverage problem with group budget constraints [11]. For the sake of completeness, we provide below the analysis of the greedy algorithm.

Let $J^{(g)}$ be the set of jobs scheduled by the greedy algorithm. For each j, let $J_j^{(g)}$ be the set of jobs scheduled by the greedy algorithm on (some resource-interval of) $\mathcal{P}_j$. Similarly, let J_j^* be the set of jobs scheduled by the optimum solution $\texttt{opt}$ on (some resource-interval of) $\mathcal{P}_j$. Partition each J_j^* into two sets: $J_{j,1}^* = J_j^* \cap J^{(g)}$ and $J_{j,2} = J_j^* - J_{j,1}^*$. The optimum profit is given by $\sum_j \texttt{profit}(J_{j,1}^*) + \sum_j \texttt{profit}(J_{j,2}^*)$. Clearly, the first sum is bounded by the profit of the greedy solution. On the other hand, our greedy choice ensures that for each j, $\texttt{profit}(J_j^{(g)}) \geq \texttt{profit}(J_{j,2}^*)$. Thus, the second sum is also bounded by the profit of the greedy solution. It follows that $\texttt{profit}(\texttt{opt}) \leq 2\texttt{profit}(J^{(g)})$.