A Near-linear Time Constant Factor Algorithm for Unsplittable Flow Problem on Line with Bag Constraints

Venkatesan T. Chakaravarthy, Anamitra R. Choudhury, and Yogish Sabharwal

IBM Research - India, New Delhi
{vechakra, anamchou, ysabharwal}@in.ibm.com

Abstract

Consider a scenario where we need to schedule a set of jobs on a system offering some resource (such as electrical power or communication bandwidth), which we shall refer to as bandwidth. Each job consists of a set (or *bag*) of job instances. For each job instance, the input specifies the start time, finish time, bandwidth requirement and profit. The bandwidth offered by the system varies at different points of time and is specified as part of the input. A feasible solution is to choose a subset of instances such that at any point of time, the sum of bandwidth requirements of the chosen instances does not exceed the bandwidth available at that point of time, and furthermore, at most one instance is picked from each job. The goal is to find a maximum profit feasible solution. We study this problem under a natural assumption called the no-bottleneck assumption (NBA), wherein the bandwidth requirement of any job instance is at most the minimum bandwidth available. We present a simple, near-linear time constant factor approximation algorithm for this problem, under NBA.

When each job consists of only one job instance, the above problem is the same as the well-studied unsplittable flow problem (UFP) on lines. A constant factor approximation algorithm is known for the UFP on line, under NBA. Our result leads to an alternative constant factor approximation algorithm for this problem. Though the approximation ratio achieved by our algorithm is inferior, it is much simpler, deterministic and faster in comparison to the existing algorithms. Our algorithm runs in near-linear time ($O(n \log^2 n)$), whereas the running time of the known algorithms is a high order polynomial. The core idea behind our algorithm is a reduction from the varying bandwidth case to the easier uniform bandwidth case, using a technique that we call *slicing*.

1998 ACM Subject Classification F.2.2 [Analysis of Algorithms]

Keywords and phrases Approximation Algorithms; Scheduling; Resource Allocation

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2010.181

1 Introduction

We consider a general resource allocation problem in which we need to schedule jobs on a system offering a certain amount of some resource (such as electrical power, processing nodes, communication bandwidth). Each job consists of a set (or *bag*) of job instances, out of which at most one can be chosen. Each job instance requires a particular amount of the resource for its execution. The total amount of the resource offered by the system is different at different points of time. Our goal is to choose a subset of job instances such that at any timeslot, the total amount of resource requirement does not exceed the total amount of the resource available at that timeslot. We wish to maximize the profit of the

IARCS Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010).
Editors: Kamal Lodaya, Meena Mahajan; pp. 181–191

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

chosen subset of jobs. The problem formulation is motivated by its applications in cloud computing, bandwidth allocation in networks, smart energy management and allocating processor nodes on a multi-processor system. We refer to [4, 11, 1, 12, 8] for a discussion on some of these applications. Motivated by such scheduling and bandwidth allocation scenarios, we study an abstract problem called the *Varying bandwidth resource allocation problem with bag constraints* (BagVBRap), introduced in [8]. We use *bandwidth* as a generic term to refer to the resource under contention. The BagVBRap problem generalizes several previously studied scheduling and resource allocation problems. We next define the problem formally.

1.1 BagVBRap Problem Definition

The input consists of a set of jobs $\mathcal{J}$. Each job $J \in \mathcal{J}$ consists of a set of *job instances* of which at most one can be selected for execution. A instance u of a job J is specified by an interval $I_u = [a, b]$, where a and b are the *start time* and the *finish time* of the instance u; we assume that a and b are integers. The job instance u is also associated with a *bandwidth requirement* ρ_u and a profit p_u. Let D be the maximum finish time over all instances so that the interval associated with every job instance is contained in the span $[1, D]$. We refer to each integer $1 \leq t \leq D$ as a *timeslot*. For each timeslot t, the input specifies a number B_t which is the *bandwidth available* at timeslot t.

We use the term *instance* as a shorthand for job instance. Let $\mathcal{U}$ denote the set of all n instances over all the jobs in $\mathcal{J}$. For each instance $u \in \mathcal{U}$, we view each interval $I_u = [a, b]$ as a set of timeslots in the range $[a, b]$. We view each job as a *bag* of its instances. We say that the instance u is *active* at a timeslot t, if $t \in I_u$. For a timeslot t, let $A(t)$ denote the set of all instances active at timeslot t.

A feasible solution is a subset of instances $S \subseteq \mathcal{U}$ such that at every timeslot t, the sum of the bandwidth requirements of the instances from S active at time t is at most B_t, i.e, for every timeslot $1 \leq t \leq D$,

$$\sum_{u \in S \cap A(t)} \rho_u \leq B_t.$$

We call this the *bandwidth constraint*. Furthermore, it is required that at most one instance is picked from each job; we call this the *bag constraint*; we view a job as a bag of instances and hence the terminology. The problem is to find a feasible solution S such that the sum of the profits of the jobs in S is maximized. This completes the problem description.

The concept of bag constraints is quite powerful. Apart from handling the notion of release time and deadline, it can also work in a more general setting where a job can specify a set of possible time intervals where it can be scheduled. Moreover, BagVBRap allows for different instances of the same job to have different bandwidth requirements, processing times and profits.

The maximum and minimum available bandwidths over all timeslots will be of use in our discussion. We denote these by $\mathrm{B}_{\max}$ and $\mathrm{B}_{\min}$, that is $\mathrm{B}_{\max} = \max_{t \in [1,D]} B_t$ and $\mathrm{B}_{\min} = \min_{t \in [1,D]} B_t$. Similarly, the maximum and minimum bandwidth requirement over all instances is also of interest. We denote these by $\rho_{\max}$ and $\rho_{\min}$. That is, $\rho_{\max} = \max_{u \in \mathcal{U}} \rho_u$ and $\rho_{\min} = \min_{u \in \mathcal{U}} \rho_u$.

Remark: We can assume that for each instance u, the start time and end time of u are in the range $[1, 2n]$, since this leaves the problem combinatorially unchanged. Thus, we can assume that $D \leq 2n$.

1.2 Prior Work

The BagVBRap problem is a generalized formulation that captures as special cases many well-studied scheduling and resource allocation problems. Here we shall describe some important special cases and then present a brief survey of some of the prior work dealing with these problems.

- *Uniform bandwidth resource allocation problem (*UBRap*):* This is the special case of the BagVBRap problem, where the bandwidth available is uniform across all timeslots and the bag constraints do not exist. Meaning, each job consists of only one instance and for all $1 \leq t \leq D$, $B_t = B$ for some fixed B given as part of the input.
- *Uniform bandwidth resource allocation problem with bag constraints (*BagUBRap*):* This is the special case of the BagVBRap problem, where the bandwidth available is uniform across all timeslots.
- *Varying bandwidth resource allocation problem (*VBRap*):* This is the special case of the BagVBRap problem, where each job has only one instance. The VBRap problem is the same as the unsplittable flow problem (UFP) on line graphs, a well-studied problem.

Calinescu et al. [7] presented a 3-approximation for the UBRap problem, based on LP rounding technique that they refer to as "listing algorithm". Independently, Bar-Noy et al. [4] also presented a local-ratio based 3-approximation algorithm for the same problem. They also show how to handle bag constraints and derive a 5-approximation algorithm for the BagUBRap problem. A further special case of the BagUBRap problem is obtained when the bandwidth requirements and the bandwidth available are all unit. This special case has been studied under the name *weighted job interval selection problem* (WJISP), for which Bar-Noy et al. [5] and, independently, Berman and Dasgupta [6] presented local-ratio based 2-approximation algorithms.

For the VBRap problem (i.e., the UFP problem on line) Chakrabarti et al. [9] presented an algorithm with an approximation ratio of $O(\log(\rho_{\max}/\rho_{\min}))$. Bansal et al. [3] presented an $O(\log n)$-approximation algorithm for the same problem. No polynomial time constant factor approximation algorithm is known for this problem. However, in a break-through result, Bansal et al. [2] obtained a quasi-PTAS for UFP on line. For the BagVBRap problem, an $O(\log(\mathrm{B}_{\max}/\mathrm{B}_{\min}))$-approximation algorithm was obtained in [8]; this was achieved by extending the LP based "listing" algorithm of Calinescu et al. [7],

Obtaining a polynomial time constant factor approximation algorithm for the VBRap problem has remained a challenging open problem. However, this has been achieved under a reasonable assumption known as the *no-bottleneck assumption* (NBA).

No Bottleneck Assumption (NBA): We say that an input to the BagVBRap problem satisfies the *no bottleneck assumption (NBA)*, if the maximum bandwidth requirement of every job instance is less than the minimum bandwidth available. That is, $\rho_{\max} \leq \mathrm{B}_{\min}$.

Chakrabarti et al. [9] obtained the first constant factor approximation algorithm for the VBRap problem, under NBA. For the same special case, Chekuri et al. [10] improved the constant factor to $(2 + \epsilon)$. Both these algorithms are based on randomized rounding of LP solutions.

1.3 Our Result and Discussion

Our main result is a constant factor approximation algorithm for the BagVBRap problem, under NBA. The running time of the algorithm is $O(n \log^2 n)$, where n is the number of job instances. The approximation ratio is 120.

An important feature of our approach is the simplicity of both the algorithm and analysis. We show how to handle the non-uniformity or the varying nature of the bandwidths available, via a reduction to the easier case of uniform bandwidths. Namely, we present a simple reduction from the BagVBRap problem with NBA to the BagUBRap problem. Given a BagVBRap input with n job instances, our algorithm produces a BagUBRap instance having at most $O(n \log n)$ job instances. We then invoke the known constant factor approximation algorithm for BagUBRap, due to Bar-Noy et al. [4], which is based on the local ratio technique, and runs in time $O(n \log n)$, where n is the number of input job instances. Thus, our algorithm for BagVBRap runs in time $O(n \log^2 n)$.

Our result yields a constant factor approximation algorithm for the UFP problem on line, with NBA. The approximation ratio is 120. As mentioned earlier, Chakrabarti et al. [9] and Chekuri et al. [10] have presented constant factor approximation algorithms for this problem. The algorithm of Chekuri et al. guarantees an approximation ratio of $(2 + \epsilon)$ (for any $\epsilon > 0$); this algorithm is based on randomized LP rounding. This algorithm offers a tradeoff between approximation ratio and running time. The best running time achievable within this framework is more than $O(n^{10})$. As the approximatio ratio approaches 2, the running time grows substantially. Though our algorithm is inferior in terms of approximation ratio, it is simpler, deterministic and runs in near-linear time ($O(n \log^2 n)$). We believe that our technique of reducing varying bandwidths to uniform bandwidths may find application in other scenarios as well.

Organization. The rest of the paper is devoted towards presenting our constant factor approximation algorithm for the BagVBRap problem with NBA. For a better exposition of our main ideas, in Section 3, we first consider a special case where the available bandwidths are all integral multiples of $\mathrm{B}_{\min}$. For this case, we present a constant factor approximation algorithm. In Section 4, we will take care of the technicalities in dealing with the general case and derive a constant factor approximation algorithm for the general case, which runs in time $O(n \log^2 n)$. Both the algorithms use reductions to BagUBRap.

2 Preliminaries

Here, we develop some notations used throughout the paper. For a set of instances $X \subseteq \mathcal{U}$ and a timeslot $t \in [1, D]$, let $\rho_X(t)$ denote the sum of bandwidth requirements of all instances in X that are active at timeslot t, i.e., $\rho_X(t) = \sum_{u \in X \cap A(t)} \rho_u$.

By a *bandwidth profile* P, we mean a function that specifies a bandwidth for each timeslot. If P and Q are two bandwidth profiles, we define $P - Q$ to be the profile R given by: $R(t) = \max\{0, P(t) - Q(t)\}$, for all timeslots t. For a bandwidth profile P and a constant c, we define $c \otimes P$ to be the profile P' given by: $P'(t) = c \cdot P(t)$, for all timeslots t. By a *uniform strip of bandwidth* b, we mean a profile which uniformly takes the bandwidth to be b, for all timeslots. We say that the profile P is an *integral profile*, if for all timeslots t, $P(t)$ is an integral multiple of $\mathrm{B}_{\min}$ (we allow $P(t) = 0$).

We say that a set of instances X *fits into a profile* P, if for all timeslots t, $\rho_X(t) \leq P(t)$. The input specifies a profile P_{in} given by: $P_{in}(t) = B_t$, for all timeslots t. We shall refer to P_{in} as the input profile. Note that a feasible solution is simply a set of instances S that fits into the input profile P_{in} and satisfies the bag constraints.

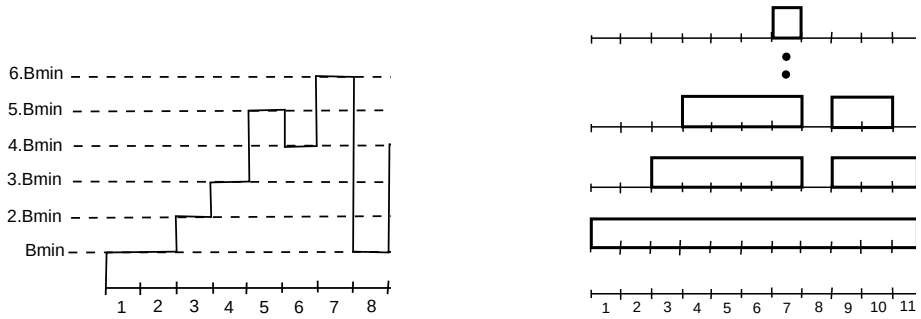

Figure 1 Illustration of slicing

3 BagVBRap with NBA: The Case of Integral Profiles

In this section, for a simple exposition of our main ideas, we focus on the special case where the input bandwidth profile P_{in} is integral. For this case, we present a constant factor approximation algorithm having running time $O(n^2 \log n)$. In the next section, we will take care of the technicalities in dealing with the non-integral input profiles and derive a constant factor approximation algorithm for the general case, which runs in time $O(n \log^2 n)$.

Let $K = \mathrm{B}_{\max}/\mathrm{B}_{\min}$; since the profile is assumed to be integral, K is an integer. We imagine the input bandwidth profile P_{in} to be a curve giving a value for each timeslot t. We slice the area under the curve horizontally into K *slices* each of height $\mathrm{B}_{\min}$, as illustrated in Figure 1. For $1 \leq j \leq K$, the bandwidth profile of the jth slice is defined as follows: for $1 \leq t \leq D$,

$$\mathrm{Slice_j(t)} \quad = \quad \begin{cases} \mathrm{B}_{\min} & \text{if } B_t \geq j \cdot \mathrm{B}_{\min} \\ 0 & \text{otherwise} \end{cases}$$

Consider a feasible solution $S \subseteq \mathcal{U}$ (which fits into the profile P_{in} and satisfies the bag constraints). The solution S is said to be *slice-respecting*, if it is possible to assign the instances of S to the K slices satisfying the following two properties: (i) each instance $u \in S$ is assigned to one of the K slices; (ii) for $1 \leq j \leq K$, the subset of instances assigned to the jth slice fits into the profile $\mathrm{Slice_j}$.

Let $S_{\mathrm{opt}} \subseteq \mathcal{U}$ be the optimum solution. Our algorithm and analysis have two main components:

- First, we will show that the optimum solution S_{opt} can be partitioned into 16 subsets such that each subset is a slice-respecting solution (see Section 3.1).
- Second, we will present an algorithm that will output a slice-respecting solution S such that the profit of S is a 5-approximation to the optimum slice-respecting solution (namely, the maximum profit solution among all slice respecting solutions). This algorithm is obtained via a reduction to the BagUBRap problem. (see Section 3.2).

It follows that S is a 80-approximation to S_{opt}. This yields the following theorem.

▶ **Theorem 1.** *There exists a* 80*-approximation algorithm for the* BagVBRap *problem with NBA when the input bandwidth profile P_{in} is integral. The running time of the algorithm is $O(n^2 \log n)$.*

3.1 Partitioning S_{opt} into Slice-respecting Solutions

In this section, we will prove the following lemma.

▶ **Lemma 2.** *Any feasible solution S can be partitioned into* 16 *subsets such that each subset is a slice-respecting solution.*

The intuitive idea behind the above claim is as follows. We will delete the first slice from the bottom of the profile and obtain a new profile. We shall identify a subset of instances Y and delete them such that the remaining instances fits into the new profile. We will show that the deleted subset of instances Y can be partitioned into a collection of 16 subsets such that each subset in the collection fits into the deleted (first) slice. We shall then apply the above procedure recursively K times obtaining K such collections. A slice-respecting solution can be formed by picking one subset from each collection. This way, we can form 16 slice respecting solutions.

The rest of the section is devoted to proving Lemma 2 formally. The following two lemmas are useful for this purpose.

▶ **Lemma 3.** *Let P be any integral profile. Let $X \subseteq \mathcal{U}$ be any subset of instances that fits into the profile P. Then, there exists a subset $Y \subseteq X$ such that: (i) for all timeslots t, $\rho_Y(t) \geq \min\{\rho_X(t), \mathrm{B}_{\min}\}$; (ii) Y fits into the uniform strip of bandwidth $4 \cdot \mathrm{B}_{\min}$.*

Proof. For a subset $T \subseteq \mathcal{U}$, a job instance is said to be *critical* for T if removal of the job instance from T causes the total bandwidth of the remaining jobs in T to fall below $\mathrm{B}_{\min}$ at some timeslot. More formally, a job instance u is said to be critical for T, if $\rho_{T\setminus\{u\}}(t) < \mathrm{B}_{\min}$ for some $t \in I_u$.

We start with $Y = X$ and then repeatedly remove jobs that are not critical for Y until no more such jobs exist. We argue that the remaining jobs in Y satisfy the required properties. The first property follows from the fact that we never remove a critical job. The second property is proved by contradiction. Suppose that for some timeslot $\tilde{t}$, $\rho_Y(\tilde{t}) > 4 \cdot \mathrm{B}_{\min}$. Let

$$t_l = \max_t\{t \leq \tilde{t} \text{ and } \rho_Y(t) < 2 \cdot \mathrm{B}_{\min}\} \quad \text{and} \quad t_r = \min_t\{t \geq \tilde{t} \text{ and } \rho_Y(t) < 2 \cdot \mathrm{B}_{\min}\}.$$

First, let us suppose that both t_l and t_r exist. Note that by definition of t_l and t_r, we have that $\rho_Y(t) \geq 2 \cdot \mathrm{B}_{\min}$, for all $t \in (t_l, t_r)$. We will now argue that there exists some job instance $u \in Y$, such that I_u is contained in (t_l, t_r). If this were not so, then for any timeslot $t' \in (t_l, t_r)$, any job instance u' active at timeslot t' would also be active at timeslot t_l or at timeslot t_r. This implies that for any $t' \in (t_l, t_r)$, $\rho_Y(t') \leq \rho_Y(t_l) + \rho_Y(t_r) < 4 \cdot \mathrm{B}_{\min}$. This would contradict our hypothesis that $\rho_Y(\tilde{t}) > 4 \cdot \mathrm{B}_{\min}$. Therefore, there exists a job instance u such that I_u is contained in (t_l, t_r). This combined with the fact that $\rho_Y(t) \geq 2 \cdot \mathrm{B}_{\min}$, for all $t \in (t_l, t_r)$ and the NBA implies that u is not a critical job. This contradicts our construction of Y.

Now, let us assume that t_l does not exist. Then $\rho_Y(t) \geq 2 \cdot \mathrm{B}_{\min}$, for all $t \in [1, \tilde{t}]$. Let $u' \in Y$ be the job with the earliest finish time and let its finish time be $t_{u'}$. Note that $\rho_Y(t) \geq 2 \cdot \mathrm{B}_{\min}$ for all $t \in [1, t_{u'}]$. This fact along with the NBA assumption implies that u' is not a critical job. This contradicts our construction of Y.

The case when t_r does not exist is handled similar to the case of t_l not existing as above. This completes the proof of the lemma. □

▶ **Lemma 4.** *Let $X \subseteq \mathcal{U}$ be any subset of instances that fits into the uniform strip of bandwidth $4 \cdot \mathrm{B}_{\min}$. Then, X can be partitioned into a collection of* 16 *subsets $\{X_1, X_2, \ldots, X_{16}\}$ such that each subset X_i fits into the uniform strip of bandwidth $\mathrm{B}_{\min}$.*

Proof. We say that an instance $u \in X$ is *large*, if $\rho_u > \mathrm{B}_{\min}/2$; otherwise, u is said to be small. Let $X_\ell \subseteq X$ be the set of large instances and let X_s be the set of small instances. Let

us first focus on the set X_ℓ. Create 8 uniform strips of bandwidth B_{min}, named $P_1, P_2, \ldots, P_8$, called *buckets*. Arrange the instances in X_ℓ in a list in increasing order of their start times. Scan this list and for each instance u, add u to a bucket where it can fit without violating the bandwidth constraint. The crucial claim is that each instance u will fit into some bucket. To see this claim, consider the first instance u that does not fit into any bucket. Let the starting timeslot of u be t_0. Since we are dealing with large instances, each bucket contains exactly one large instance active at the timeslot t_0 . Let Y be the set of these instances. Their combined bandwidth at t_0 is $\rho_Y(t_0) > 4 \cdot B_{min}$. This is a contradiction since X is assumed to fit into a uniform strip of bandwidth $4 \cdot B_{min}$.

The argument for small tasks is similar. Consider 8 uniform strips of bandwidth B_{min}, Q_1, Q_2, ..., Q_8, which are referred to as buckets. Scan the small instances in the increasing order of their start times. For each instance u, add u to a bucket where it would fit without violating the bandwidth constraint. As before, we claim that every instance will fit into at least one bucket. To see this, suppose u be the first instance that does not fit into any bucket. Let the starting timeslot of u be t_0. For each $1 \leq i \leq 8$, let Y_i be the set of instances in Q_i active at timeslot t_0. Since u is small, $\rho_{Y_i}(t_0) > B_{min}/2$. Let Y be the union of $Y_1, Y_2, \ldots, Y_8$. Their combined bandwidth at t_0 is $\rho_Y(t_0) > 4 \cdot B_{min}$. This is a contradiction since X is assumed to fit into a uniform strip of bandwidth $4 \cdot B_{min}$.

The set of instances added to the 16 buckets $P_1, P_2, \ldots, P_8$ and $Q_1, Q_2, \ldots, Q_8$ are taken to be the required subsets $X_1, X_2, \ldots, X_{16}$. □

We now prove Lemma 2. Let $P_0 = P_{in}$ be the input profile and let $X_0 = S$ be the given solution. We first invoke Lemma 3 with P_0 and X_0 as inputs and obtain a set of instances Y_1. We then apply Lemma 4 to partition Y_1 into a collection of 16 subsets $\{Y_1^1, Y_1^2, \ldots, Y_1^{16}\}$. Note that each of these subsets Y_1^i fits into the profile of the first slice. We delete the first slice (bottom-most slice) from the profile P_0 and obtain a profile P_1 (formally, we set $P_1 = P_0 - \text{Slice}_1$). We also delete the set of instances Y_1 from X_0 and get a new set of instances $X_1 = X_0 - Y_1$. Observe that the set of instances X_1 fits into the profile P_1. This allows us to apply the above process starting with P_1 and X_1 as inputs. Overall, we will apply the above process iteratively K times.

Formally, for $j = 1$ to K, we do as follows:

- Invoke Lemma 3 with P_{j-1} and X_{j-1} as inputs and obtain a set of instances Y_j.
- Invoke Lemma 4 to partition Y_j into a collection of 16 subsets $\{Y_j^1, Y_j^2, \ldots, Y_j^{16}\}$. Note that each set Y_j^i fits into the profile of the jth slice.
- Define a new integral profile $P_j = P_{j-1} - \text{Slice}_j$.
- Define $X_j = X_{j-1} - Y_j$. The set of instances X_j fits into the profile P_j.

We now form 16 solutions: for $1 \leq i \leq 16$, let $Z_i = \cup_{j=1}^{K} Y_j^i$. Each set Z_i is a slice-respecting solution. This completes the proof of Lemma 2.

3.2 Approximating the Optimum Slice-respecting Solution

In this section, we shall present an algorithm for finding a 5-approximation to the optimum slice-respecting solution. We achieve this goal via a reduction to the BagUBRap problem, for which Bar-Noy et al. [4] designed a local-ratio based 5-approximation algorithm.

Let S^* be the optimum slice-respecting solution. Our goal is to find a 5-approximation to S^*, via a reduction to the BagUBRap problem. Let $\mathcal{I}$ be the input instance, which we will transform into an input instance $\widetilde{\mathcal{I}}$ of BagUBRap. Let the timespan of $\mathcal{I}$ be $[1, D]$ so that all the job instances finish before D. In the transformed instance $\widetilde{\mathcal{I}}$, the timespan will be $[1, KD]$, where K is the number of slices. The bandwidth profile of $\widetilde{\mathcal{I}}$ is obtained by concatenating

the bandwidth profiles of the K slices; namely, for $1 \leq j \leq K$, the range $[1+(j-1)D, jD]$ corresponds to the jth slice. Formally, in the input instance $\widetilde{I}$, the bandwidth available $\widetilde{B}_t$ is declared as follows: for $1 \leq j \leq K$ and for each timeslot $t \in [1+(j-1)D, jD]$, we set $\widetilde{B}_t = \mathrm{Slice_j(t-(j-1)D)}$. Note that in $\widetilde{\mathcal{I}}$, the bandwidths available at all timeslots are either $\mathrm{B_{min}}$ or 0.

For each job $J \in \mathcal{J}$ of the input $\mathcal{I}$, we create a corresponding job $\widetilde{J}$ in $\widetilde{\mathcal{I}}$. For each job instance $u \in \mathcal{U}$ in the input instance $\mathcal{I}$ with associated interval $I_u = [a, b]$, we create at most K copies of u as follows. For each slice $1 \leq j \leq K$, we create a copy of u, if the instance u can be scheduled in the jth slice (meaning, for all $t \in [a, b]$, $\mathrm{Slice_j(t)} \neq 0$); this copy of u is declared to have the interval $I_u^j = [(j-1)D + a, (j-1)D + b]$. This way, at most K copies are created for the instance u. Each of these copies will have the same bandwidth requirement and profit as the instance u. If $J \in \mathcal{J}$ is the job to which the instance u belongs, then all these copies of u are made instances of the corresponding job $\widetilde{J}$ in the transformed input $\widetilde{\mathcal{I}}$. For instance, if a job $J \in \mathcal{J}$ had d instances in $\mathcal{I}$, its corresponding job in $\widetilde{\mathcal{I}}$ would have at most dK instances.

A minor issue in the above construction is that in $\widetilde{\mathcal{I}}$, the bandwidths available are not uniform across all the timeslots. However, these are all either $\mathrm{B_{min}}$ or 0. This is easily addressed via compressing the profile by deleting all timeslots where the bandwidth availability is 0. This way, we get a BagUBRap instance. This completes the reduction.

It is easy to see that a slice-respecting feasible solution S to $\mathcal{I}$ can be converted into a feasible solution $\widetilde{S}$ for $\widetilde{\mathcal{I}}$ and vice versa. To see the forward direction, if u is an instance picked in the solution S and assigned to slice j, we pick the instance I_u^j in $\widetilde{S}$. For the reverse direction, if the instance I_u^j is picked in the solution $\widetilde{S}$, we include the instance I_u in S and assign it to the slice j. Now invoking the 5-approximation algorithm for the BagUBRap problem [4], we get the desired 5-approximation to the optimum slice-respecting solution.

The 5-approximation algorithm for the BagUBRap problem runs in time $O(\widetilde{n}\log(\widetilde{n}))$, where $\widetilde{n}$ is the number of job instances in $\widetilde{\mathcal{I}}$. In our reduction, for each job instance $u \in \mathcal{I}$, at most K instances are created in $\widetilde{\mathcal{I}}$ and hence, $\widetilde{n} \leq Kn$. Recall that $K = \mathrm{B_{max}}/\mathrm{B_{min}}$. We can assume that $\mathrm{B_{max}} \leq n\rho_{\max}$. Under NBA, we also have that $\mathrm{B_{min}} \geq \rho_{\max}$. Hence, $K = O(n)$. Thus, our algorithm runs in time $O(n^2 \log n)$.

4 A Constant Factor Approximation for the General Case

We now consider the general case involving non-integral input profiles and derive a constant factor approximation algorithm having running time $O(n \log^2 n)$. The core idea of slicing is the same as that of the earlier section. The new result is obtained by dealing with some technicalities. First, instead of slicing the input profile into slices of equal height, here we shall slice the profile into slices of geometrically increasing heights. This enables us to bring down the running time. Secondly, we shall "floor" the input profile by carefully decreasing the bandwidth available at every timeslot so that it becomes "geometric" and it is therefore, suitable for "geometric slicing". However, the "flooring" increases the approximation ratio by a constant factor.

We say that a profile P is a *geometric integral profile* (GIP), if one of the following two conditions is true:

- For all timeslots t, either $P(t) = 0$ or $P(t) = 2^i \cdot \mathrm{B_{min}}$, for some integer $i \geq 0$.
- There exists an integer constant $c \geq 0$ such that for all timeslots t, either $P(t) = 0$ or $P(t) = (2^i - 2^c)\mathrm{B_{min}}$, for some integer $i \geq 0$.

In the former case the profile is said to be a *type-1* GIP and in the latter case, the profile

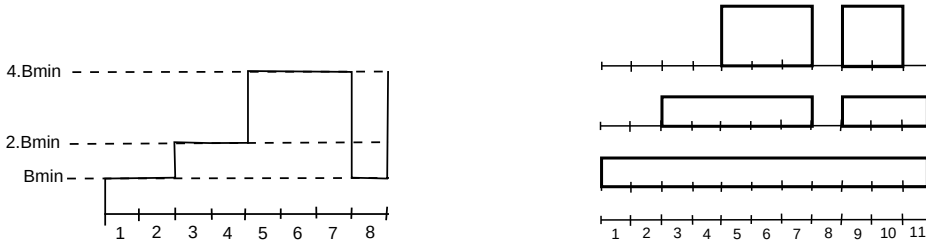

Figure 2 Illustration of slicing

said to be a *type-2* GIP. The idea of having two different types of GIPs is that we shall start with a type-1 GIP and as we delete slices iteratively, we obtain type-2 GIPs.

We first convert the input profile P_{in} into a type-1 GIP by "flooring" the bandwidths, as follows. Define a new profile $\widetilde{P}_{in}$ given by $\widetilde{P}_{in}(t) = 2^i \cdot \mathrm{B}_{\min}$, where $i = \lfloor \log(P_{in}(t)/\mathrm{B}_{\min}) \rfloor$. Figure 2 presents the flooring of the profile shown in Figure 1. A feasible solution $S \subseteq \mathcal{U}$ (which fits into the profile P_{in}) may not fit into the profile $\widetilde{P}_{in}$. Our algorithm will actually output a solution that fits into the profile $\widetilde{P}_{in}$. In this process, we lose a constant factor in the approximation.

Similar to the previous section, we now define the notion of slices and slice-respecting solutions. Let $K = 1 + \lfloor \log(\mathrm{B}_{\max}/\mathrm{B}_{\min}) \rfloor$. We slice the profile $\widetilde{P}_{in}$ horizontally into K *slices* in a geometric manner. The profile of the first slice Slice_1 is the uniform strip of bandwidth $\mathrm{B}_{\min}$. For $2 \leq j \leq K$, the bandwidth profile of the jth slice is given by the following profile: for $1 \leq t \leq D$,

$$\mathrm{Slice_j(t)} \quad = \quad \begin{cases} 2^{j-2} \cdot \mathrm{B}_{\min} & \text{if } \widetilde{P}_{in}(t) \geq 2^{j-1} \cdot \mathrm{B}_{\min} \\ 0 & \text{otherwise} \end{cases}$$

See Figure 2 for an illustration of the slices.

Consider a feasible solution $S \subseteq \mathcal{U}$ (which fits into the profile P_{in} and satisfies the bag constraints). The solution S is said to be *slice-respecting*, if it is possible to assign the instances of S to the K slices satisfying the following two properties: (i) each instance $u \in S$ is assigned to one of the K slices; (ii) for $1 \leq j \leq K$, the subset of instances assigned to the jth slice fits into the profile $\mathrm{Slice_j}$.

Let $S_{\mathrm{opt}} \subseteq \mathcal{U}$ be the optimum solution. Our algorithm and analysis have two main components:

- First, we will show that the solution S_{opt} can be partitioned into 24 subsets such that each subset is a slice-respecting solution.
- Second, we will present an algorithm that will output a slice-respecting solution S such that the profit of S is a 5-approximation to the optimum slice-respecting solution.

It follows that S is a 120-approximation to S_{opt}. This yields the following theorem.

▶ **Theorem 5.** *There exists a* 120*-approximation algorithm for the* BagVBRap *problem with NBA. The running time of the algorithm is* $O(n \log^2 n)$.

We now focus on the first component and prove a lemma similar in spirit to Lemma 2.

▶ **Lemma 6.** *Any feasible solution* $S \subseteq \mathcal{U}$ *can be partitioned into* 24 *subsets such that each subset is a slice-respecting solution.*

We first state two variants of Lemma 3 that deal with the two types of GIPs. The proofs are similar to that of Lemma 3.

▶ **Lemma 7.** *Let P be any type-1* GIP*. Let $X \subseteq \mathcal{U}$ be any subset of instances that fits into the profile $2 \otimes P$. Then, there exists $Y \subseteq X$ such that: (i) for all timeslots t, $\rho_Y(t) \geq \min\{\rho_X(t), 2 \cdot \mathrm{B_{min}}\}$; (ii) Y fits the uniform strip of bandwidth $6 \cdot \mathrm{B_{min}}$.*

▶ **Lemma 8.** *Let P be any type-2* GIP *with parameter $c \geq 0$. Let $X \subseteq \mathcal{U}$ be any subset of instances that fits into the profile $2 \otimes P$. Then, there exists $Y \subseteq X$ such that: (i) for all timeslots t, $\rho_Y(t) \geq \min\{\rho_X(t), 2 \cdot 2^c \cdot \mathrm{B_{min}}\}$; (ii) Y fits into the uniform strip of bandwidth $6 \cdot 2^c \cdot \mathrm{B_{min}}$.*

We next state a variant of Lemma 4. The proof is similar to that of Lemma 4.

▶ **Lemma 9.** *Let $X \subseteq \mathcal{U}$ be any subset of instances that fits into the uniform strip of bandwidth $6 \cdot \alpha \cdot \mathrm{B_{min}}$, for some integer $\alpha \geq 1$. Then, X can be partitioned into 24 subsets $X_1, X_2, \ldots, X_{24}$ such that each subset X_i fits into the uniform strip of bandwidth $\alpha \cdot \mathrm{B_{min}}$.*

We now prove Lemma 6. The proof is similar to that of Lemma 2 and is proved in an iterative manner. In the first iteration, we apply Lemma 7 and Lemma 9 on profile $\widetilde{P}_{in}$, which is a type-1 GIP. In the subsequent iterations, we will apply Lemma 8 and 9, as the profiles considered in these iterations are type-2 GIPs.

Let $P_0 = \widetilde{P}_{in}$ be the input profile, which is a type-1 GIP. Let $X_0 = S$ be the given solution that fits into the profile P_{in}. Observe that X_0 fits into the profile $2 \otimes P_0$. We first invoke Lemma 7 with P_0 and X_0 as inputs and obtain a set of instances Y_1. We then apply Lemma 9 to partition Y_1 into 24 subsets $Y_1^1, Y_1^2, \ldots, Y_1^{24}$ (here $\alpha = 1$). Note that each of these subsets Y_1^i fits into the profile Slice_1. We delete the first slice (bottom-most slice) from the profile P_0 and obtain a profile P_1 (formally, we set $P_1 = P_0 - \mathrm{Slice}_1$). We also delete the set of instances Y_1 from X_0 and get a new set of instances $X_1 = X_0 - Y_1$.

The profile P_1 is a type-2 GIP with parameter $c = 0$ and X_1 fits into $2 \otimes P_1$. We invoke Lemma 8 with P_1 and X_1 as inputs and obtain a set of instances Y_2. We then apply Lemma 9 to partition Y_2 into 24 subsets $Y_2^1, Y_2^2, \ldots, Y_2^{24}$ (here $\alpha = 2^0 = 1$). Note that each of these subsets Y_2^i fits into the profile Slice_2. We now delete the second slice from the profile P_1 and obtain a profile P_2 (formally, we set $P_2 = P_1 - \mathrm{Slice}_2$). We also delete the set of instances Y_2 from X_1 and get a new set of instances $X_2 = X_1 - Y_2$. The profile P_2 is a type-2 GIP with parameter $c = 1$ and X_2 fits into $2 \otimes P_2$.

We can now iteratively apply the above process starting with P_2 and X_2 as inputs. We continue like this for K iterations. Formally, for $j = 2$ to K, do as follows:

- Invoke Lemma 8 with P_{j-1} and X_{j-1} as inputs and obtain a set of instances Y_j.
- Invoke Lemma 9 to partition Y_j into 24 subsets $Y_j^1, Y_j^2, \ldots, Y_j^{24}$. Each set Y_j^i fits into the profile $\mathrm{Slice_j}$.
- Define a profile P_j given by $P_j = P_{j-1} - \mathrm{Slice_j}$. The profile P_j is a type-2 GIP with parameter $c = j - 2$.
- Define $X_j = X_{j-1} - Y_j$. The set of instances X_j fits into the profile $2 \otimes P_j$.

We now form 24 solutions: for $1 \leq i \leq 24$, let $Z_i = \cup_{j=1}^{K} Y_j^i$. Each set Z_i is a slice-respecting solution. This completes the proof of Lemma 6.

We now outline the second component required to prove Theorem 5. What we need is an algorithm that approximates the optimum slice respecting solution within a factor of 5. The construction is the same as that of Section 3.2. However, we do not get a BagUBRap instance, since the bandwidths available are not uniform. Nevertheless, we can scale the

slices (and the associated job instances), so as to make all the slices of uniform bandwidth. This way we can get a BagUBRap instance.

In the current scenario $K = O(\log(B_{max}/B_{min}))$. Now an analysis similar to the one performed in the previous section shows that the running time of our algorithm is $O(n \log^2 n)$.

References

1. S. Albers. Resource Management in Large Networks. In J. Lerner, D. Wagner, and K. Zweig, editors, *Algorithmics for Large and Complex Networks: Design, Analysis, and Simulation*, pages 227–246. Springer Berlin/Heidelberg, 2009.
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 (STOC)*, pages 721–729, 2006.
3. N. Bansal, Z. Friggstad, R. Khandekar, and M. Salavatipour. A logarithmic approximation for unsplittable flow on line graphs. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 702–709, 2009.
4. 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.
5. A. Bar-Noy, S. Guha, , J. Noar, and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *Siam Journal of Computing*, 31(2):331–352, 2001.
6. P. Berman and B. Dasgupta. Multi-phase algorithms for throughput maximization for real-time scheduling. *Journal of Combinatorial Optimization*, 4:307–323, 2000.
7. 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.
8. V. Chakaravarthy, V. Pandit, Y. Sabharwal, and D. Seetharam. Varying bandwidth resource allocation problem with bag constraints. In *24th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 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. T. Erlebach and F. Spieksma. Interval selection: Applications, algorithms, and lower bounds. *Journal of Algorithms*, 46(1):27–53, 2003.
12. M. Flammini, G. Monaco, G. Moscardelli, H. Shachnai, M. Shalom, T. Tamir, and S. Zaks. Minimizing total busy time in parallel scheduling with application to optical networks. In *23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2009.