



# Stochastic Makespan Minimization in Structured Set Systems (Extended Abstract)

Anupam Gupta<sup>1</sup>, Amit Kumar<sup>2</sup>, Viswanath Nagarajan<sup>3(✉)</sup>,  
and Xiangkun Shen<sup>4</sup>

<sup>1</sup> Carnegie Mellon University, Pittsburgh, USA

<sup>2</sup> Indian Institute of Technology, Delhi, Delhi, India

<sup>3</sup> University of Michigan, Ann Arbor, USA  
viswa@umich.edu

<sup>4</sup> Yahoo! Research, New York City, USA

**Abstract.** We study stochastic combinatorial optimization problems where the objective is to minimize the expected maximum load (a.k.a. the makespan). In this framework, we have a set of  $n$  tasks and  $m$  resources, where each task  $j$  uses some subset of the resources. Tasks have random sizes  $X_j$ , and our goal is to non-adaptively select  $t$  tasks to minimize the expected maximum load over all resources, where the load on any resource  $i$  is the total size of all selected tasks that use  $i$ . For example, given a set of intervals in time, with each interval  $j$  having random load  $X_j$ , how do we choose  $t$  intervals to minimize the expected maximum load at any time? Our technique is also applicable to other problems with some geometric structure in the relation between tasks and resources; e.g., packing paths, rectangles, and “fat” objects. Specifically, we give an  $O(\log \log m)$ -approximation algorithm for all these problems.

Our approach uses a strong LP relaxation using the cumulant generating functions of the random variables. We also show an LP integrality gap of  $\Omega(\log^* m)$  even for the problem of selecting intervals on a line.

## 1 Introduction

Consider the following task scheduling problem: an event center receives requests/tasks from its clients. Each task  $j$  specifies a start and end time (denoted  $(a_j, b_j)$ ), and the amount  $x_j$  of some shared resource (e.g., staff support) that this task requires throughout its duration. The goal is to accept some target  $t$  number of tasks so that the maximum resource-utilization over time is as small as possible. Concretely, we want to choose a set  $S$  of tasks with  $|S| = t$  to minimize

$$\max_{\text{times } \tau} \underbrace{\sum_{j \in S: \tau \in [a_j, b_j]} x_j}_{\text{usage at time } \tau}$$

All missing proofs and details can be found in the full version [11].

© Springer Nature Switzerland AG 2020

D. Bienstock and G. Zambelli (Eds.): IPCO 2020, LNCS 12125, pp. 158–170, 2020.

[https://doi.org/10.1007/978-3-030-45771-6\\_13](https://doi.org/10.1007/978-3-030-45771-6_13)

This can be modeled as an interval packing problem: if the sizes are identical, the natural LP is totally unimodular and we get an optimal algorithm. For general sizes, there is a constant-factor approximation algorithm [3].

However, in many settings, we may not know the resource consumption  $X_j$  precisely up-front, at the time we need to make a decision. Instead, we may be only given estimates. What if the requirement  $X_j$  is a random variable whose distribution is given to us? Again we want to choose  $S$  of size  $t$ , but this time we want to minimize the *expected* maximum usage:

$$\mathbb{E} \left[ \max_{\text{times } \tau} \sum_{j \in S: \tau \in [a_j, b_j]} X_j \right].$$

Note that our decision to pick task  $j$  affects all times in  $[a_j, b_j]$ , and hence the loads on various places are no longer independent: how can we effectively reason about such a problem?

In this paper we consider general resource allocation problems of the following form. There are several tasks and resources, where each task  $j$  has some size  $X_j$  and uses some subset  $U_j$  of resources. That is, if task  $j$  is selected then it induces a load of  $X_j$  on every resource in  $U_j$ . Given a target  $t$ , we want to select a subset  $S$  of  $t$  tasks to minimize the *expected maximum load* over all resources. For the non-stochastic versions of these problems (when  $X_j$  is a single value and not a random variable), we can use the natural linear programming (LP) relaxation and randomized rounding to get an  $O(\frac{\log m}{\log \log m})$ -approximation algorithm; here  $m$  is the number of resources. However, much better results are known when the task-resource incidence matrix has some geometric structure. One such example appeared above: when the resources have some linear structure, and the tasks are intervals. Other examples include selecting rectangles in a plane (where tasks are rectangles and resources are points in the plane), and selecting paths in a tree (tasks are paths and resources are edges/vertices in the tree). This class of problems has received a lot of attention and has strong approximation guarantees, see e.g. [1–8].

However, the *stochastic* counterparts of these resource allocation problems remain wide open. Can we achieve good approximation algorithms when the task sizes  $X_j$  are random variables? We refer to this class of problems as stochastic makespan minimization (GENMAKESPAN). In the rest of this work, we assume that the distributions of all the random variables are known, and that the r.v.s  $X_j$ s are independent.

## 1.1 Results and Techniques

We show that good approximation algorithms are indeed possible for GENMAKESPAN problems that have certain geometric structure. We consider the following two assumptions:

- *Deterministic problem assumption:* There is a linear-program based  $\alpha$  approximation algorithm for a suitable deterministic variant of GENMAKESPAN.

- *Well-covered assumption:* for any subset  $D \subseteq [m]$  of resources and tasks  $L(D)$  incident to  $D$ , the tasks in  $L(D)$  incident to any resource  $i \in [m]$  are “covered” by at most  $\lambda$  resources in  $D$ .

These assumptions are formalized in Sect. 2. To give some intuition for these assumptions, consider intervals on the line. The first assumption holds by the results of [3]. The second assumption holds because each resource is some time  $\tau$ , and the tasks using time  $\tau$  can be covered by two resources in  $D$ , namely times  $\tau_1, \tau_2 \in D$  such that  $\tau_1 \leq \tau \leq \tau_2$ .

**Theorem 1 (Main (Informal)).** *There is an  $O(\alpha\lambda \log \log m)$ -approximation algorithm for stochastic makespan minimization (GENMAKESPAN), with  $\alpha$  and  $\lambda$  as in the above assumptions.*

We also show that both  $\alpha$  and  $\lambda$  are constant in a number of geometric settings: for intervals on a line, for paths in a tree, for rectangles in a plane and for “fat” objects (such as disks) in a plane. Therefore, we obtain an  $O(\log \log m)$ -approximation algorithm in all these cases.

A first naive approach for GENMAKESPAN is (i) to write an LP relaxation with expected sizes  $\mathbb{E}[X_j]$  as deterministic sizes and then (ii) to use any LP-based  $\alpha$ -approximation algorithm for the deterministic problem. However, this approach only yields an  $O(\alpha \frac{\log m}{\log \log m})$  approximation ratio, due to the use of union bounds in calculating the expected maximum. Our idea is to use the structure of the problem to improve the approximation ratio.

Our approach is as follows. First, we use the (scaled) logarithmic moment generating function (log-mgf) of the random variables  $X_j$  to define deterministic surrogates to the random sizes. Second, we formulate a strong LP relaxation with an exponential number of “volume” constraints that use the log-mgf values. These two ideas were used earlier for stochastic makespan minimization in settings where each task loads a single resource [10, 14]. In the example above, where each task uses only a single time instant. However, we need a more sophisticated LP for GENMAKESPAN to be able to handle the combinatorial structure when tasks use many resources. Despite the large number of constraints, this LP can be solved approximately in polynomial time, using the ellipsoid method and using a maximum-coverage algorithm as the separation oracle. Third (and most important), we provide an iterative-rounding algorithm that partitions the tasks/resources into  $O(\log \log m)$  many nearly-disjoint instances of the deterministic problem. The analysis of our rounding algorithm relies on both the assumptions above, and also on the volume constraints in our LP and on properties of the log-mgf.

We also show some limitations of our approach. For GENMAKESPAN involving intervals in a line (which is our simplest application), we prove that the integrality gap of our LP is  $\Omega(\log^* m)$ . This rules out a constant-factor approximation via this LP. For GENMAKESPAN on more general set-systems (without any structure), we prove that the integrality gap can be  $\Omega(\frac{\log m}{(\log \log m)^2})$  even if all deterministic instances solved in our algorithm have an  $\alpha = O(1)$  integrality gap. This suggests that we do need to exploit additional structure—such

as the well-covered assumption above—in order to obtain significantly better approximation ratios via our LP.

## 1.2 Related Work

The deterministic counterparts of the problems studied here are well-understood. In particular, there are LP-based  $O(1)$ -approximation algorithms for intervals in a line [3], paths in a tree (with edge loads) [8] and rectangles in a plane (under some restrictions) [1].

Our techniques draw on prior work on stochastic makespan minimization for identical [14] and unrelated [10] resources; but there are also important new ideas. In particular, the use of log-mgf values as the deterministic proxy for random variables comes from [14] and the use of log-mgf values at multiple scales comes from [10]. The “volume” constraints in our LP also has some similarity to those in [10]: however, a key difference here is that the random variables loading different resources are correlated (whereas they were independent in [10]). Indeed, this is why our LP can only be solved approximately whereas the LP relaxation in [10] was optimally solvable. We emphasize that our main contribution is the rounding algorithm, which uses a new set of ideas; these lead to the  $O(\log \log m)$  approximation bound, whereas the rounding in [10] obtained a constant-factor approximation. Note that we also prove a super-constant integrality gap in our setting, even for the case of intervals in a line.

The stochastic load balancing problem on unrelated resources has also been studied for general  $\ell_p$ -norms (note that the makespan corresponds to the  $\ell_\infty$ -norm) and a constant-factor approximation is known [15]. We do not consider  $\ell_p$ -norms in this paper.

## 2 Problem Definition and Preliminaries

We are given  $n$  tasks and  $m$  resources. Each task  $j \in [n]$  uses some subset  $U_j \subseteq [m]$  of resources. For each resource  $i \in [m]$ , define  $L_i \subseteq [n]$  to be the tasks that utilize  $i$ . Each task  $j \in [n]$  has a *random* size  $X_j$ . If a task  $j$  is selected into our set  $S$ , it adds a load of  $X_j$  to each resource in  $U_j$ : the load on resource  $i \in [m]$  is  $Z_i := \sum_{j \in S \cap L_i} X_j$ . The makespan is the maximum load, i.e.  $\max_{i=1}^m Z_i$ . The goal is to select a subset  $S \subseteq [n]$  with  $t$  tasks to minimize the expected *makespan*:

$$\min_{S \subseteq [n]: |S|=t} \mathbb{E} \left[ \max_{i=1}^m \sum_{j \in S \cap L_i} X_j \right]. \quad (1)$$

The distribution of each r.v.  $X_j$  is known (we use this knowledge only to compute some “effective” sizes below), and these distributions are independent.

For any subset  $K \subseteq [m]$  of resources, let  $L(K) := \cup_{i \in K} L_i$  be the set of tasks that utilize at least one resource in  $K$ .

## 2.1 Structure of Set Systems: The Two Assumptions

Our results hold when the following two properties are satisfied by the set system  $([n], \mathcal{L})$ , where  $\mathcal{L}$  is the collection of sets  $L_i$  for each  $i \in [n]$ . Note that the set system has  $n$  elements (corresponding to tasks) and  $m$  sets (corresponding to resources).

**A1 ( $\alpha$ -packable):** A set system  $([n], \mathcal{L})$  is said to be  $\alpha$ -packable if for any assignment of size  $s_j \geq 0$  and reward  $r_j \geq 0$  to each element  $j \in [n]$ , and any threshold parameter  $\theta \geq \max_j s_j$ , there is a polynomial-time algorithm that rounds a fractional solution  $y$  to the following LP relaxation into an integral solution  $\hat{y}$ , losing a factor of at most  $\alpha \geq 1$ . (I.e.,  $\sum_j r_j \hat{y}_j \geq \frac{1}{\alpha} \sum_j r_j y_j$ .)

$$\max \left\{ \sum_{j \in [n]} r_j \cdot y_j : \sum_{j \in L} s_j \cdot y_j \leq \theta, \forall L \in \mathcal{L}, \text{ and } 0 \leq y_j \leq 1, \forall j \in [n] \right\}. \quad (2)$$

We also assume that the support of  $\hat{y}$  is contained in the support of  $y$ .<sup>1</sup>

**A2 ( $\lambda$ -safe):** Let  $[m]$  be the indices of the sets in  $\mathcal{L}$ ; recall that these are the resources. The set system  $([n], \mathcal{L})$  is  $\lambda$ -safe if for every subset  $D \subseteq [m]$  of (“dangerous”) resources, there exists a subset  $M \supseteq D$  of (“safe”) resources, such that (a)  $|M|$  is polynomially bounded by  $|D|$  and moreover, (b) for every  $i \in [n]$ , there is a subset  $R_i \subseteq M$ ,  $|R_i| \leq \lambda$ , such that  $L_i \cap L(D) \subseteq L(R_i)$ . Recall that  $L(D) = \cup_{h \in D} L_h$ . We denote the set  $M$  as  $\text{Extend}(D)$ .

Let us give an example. Suppose  $P$  consists of  $m$  points on the real line, and consider  $n$  intervals  $I_1, \dots, I_n$  on the line. The set system is defined on  $n$  elements (one for each interval), with  $m$  sets with the set  $L_i$  for point  $i \in P$  containing the indices of intervals that contain  $i$ . The  $\lambda$ -safe condition says that for any subset  $D$  of points in  $P$ , we can find a superset  $M$  which is not much larger such that for any point  $i \in P$ , there are  $\lambda$  points in  $M$  containing all the intervals that pass through both  $i$  and  $D$ . In other words, if these intervals contribute any load to  $i$  and  $D$ , they also contribute to one of these  $\lambda$  points. And indeed, choosing  $M = D$  ensures that  $\lambda = 2$ : for any  $i$  we choose the nearest points in  $M$  on either side of  $i$ .

Other families that are  $\alpha$ -packable and  $\lambda$ -safe include:

- Each element in  $[n]$  corresponds to a path in a tree, with the set  $L_i$  being the subset of paths through node  $i$ .
- Elements in  $[n]$  correspond to rectangles or disks in a plane, and each  $L_i$  consists of rectangles/disks containing a particular point  $i$  in the plane.

For a subset  $X \subseteq [n]$ , the projection of  $([n], \mathcal{L})$  to  $X$  is the smaller set system  $([n], \mathcal{L}|_X)$ , where  $\mathcal{L}|_X = \{L \cap X \mid L \in \mathcal{L}\}$ . The following lemma formalizes that packability and safeness properties also hold for sub-families and disjoint unions.

<sup>1</sup> The support of vector  $z \in \mathbb{R}_+^n$  is  $\{j \in [n] : z_j > 0\}$  which corresponds to its positive entries.

**Lemma 1.** Consider a set system  $([n], \mathcal{L})$  that is  $\alpha$ -packable and  $\lambda$ -safe. Then,

- (i) for all  $X \subseteq [n]$ , the set system  $(X, \mathcal{L})$  is  $\alpha$ -packable and  $\lambda$ -safe, and
- (ii) given a partition  $X_1, \dots, X_s$  of  $[n]$ , and set systems  $(X_1, \mathcal{L}_1), \dots, (X_s, \mathcal{L}_s)$ , where  $\mathcal{L}_i = \mathcal{L}|_{X_i}$ , the disjoint union of these systems is also  $\alpha$ -packable.

We consider the GENMAKESPAN problem for settings where the set system  $([n], \{L_i\}_{i \in [m]})$  is  $\alpha$ -packable and  $\lambda$ -safe for small parameters  $\alpha$  and  $\lambda$ .

**Theorem 2 (Main Result).** For any instance of GENMAKESPAN where the corresponding set system  $([n], \{L_i\}_{i \in [m]})$  is  $\alpha$ -packable and  $\lambda$ -safe, there is an  $O(\alpha\lambda \cdot \log \log m)$ -approximation algorithm.

### 2.2 Effective Size and Random Variables

In all the arguments that follow, imagine that we have scaled the instance so that the optimal expected makespan is between  $\frac{1}{2}$  and 1. It is useful to split each random variable  $X_j$  into two parts:

- the *truncated* random variable  $X'_j := X_j \cdot \mathbf{I}_{(X_j \leq 1)}$ , and
- the *exceptional* random variable  $X''_j := X_j \cdot \mathbf{I}_{(X_j > 1)}$ .

These two kinds of random variables behave very differently with respect to the expected makespan. Indeed, the expectation is a good measure of the load due to exceptional r.v.s, whereas one needs a more nuanced notion for truncated r.v.s (as we discuss below). The following result was shown in [14]:

**Lemma 2 (Exceptional Items Lower Bound).** Let  $\{X''_j\}$  be non-negative discrete random variables each taking value zero or at least  $L$ . If  $\sum_j \mathbb{E}[X''_j] \geq L$  then  $\mathbb{E}[\max_j X''_j] \geq L/2$ .

We now consider the trickier case of truncated random variables  $X'_j$ . We want to find a deterministic quantity that is a good surrogate for each random variable, and then use this deterministic surrogate instead of the actual random variable. For stochastic load balancing, a useful surrogate is the *effective size*, which is based on the logarithm of the (exponential) moment generating function (also known as the cumulant generating function) [9, 10, 12, 13].

**Definition 1 (Effective Size).** For any r.v.  $X$  and integer  $k \geq 2$ , define

$$\beta_k(X) := \frac{1}{\log k} \cdot \log \mathbb{E}\left[e^{(\log k) \cdot X}\right]. \tag{3}$$

Also define  $\beta_1(X) := \mathbb{E}[X]$ .

To see the intuition for the effective size, consider a set of independent r.v.s  $Y_1, \dots, Y_k$  all assigned to the same resource. The following lemma, whose proof is very reminiscent of the standard Chernoff bound (see [12]), says that the load is not much higher than the expectation.

**Lemma 3 (Effective Size: Upper Bound).** For indep. r.v.s  $Y_1, \dots, Y_n$ , if  $\sum_i \beta_k(Y_i) \leq b$  then  $\Pr[\sum_i Y_i \geq c] \leq \frac{1}{k^{c-b}}$ .

The usefulness of the effective size comes from a partial converse [14]:

**Lemma 4 (Effective Size: Lower Bound).** Let  $X_1, X_2, \dots, X_n$  be independent  $[0, 1]$  r.v.s, and  $\{\widetilde{L}_i\}_{i=1}^m$  be a partition of  $[n]$ . If  $\sum_{j=1}^n \beta_m(X_j) \geq 17m$  then

$$\mathbb{E} \left[ \max_{i=1}^m \sum_{j \in \widetilde{L}_i} X_j \right] = \Omega(1).$$

### 3 The General Framework

In this section we provide our main algorithm, which is used to prove Theorem 2. The idea is to write a suitable LP relaxation for the problem (using the effective sizes as deterministic surrogates for the stochastic jobs), to solve this exponentially-sized LP, and then to round the solution. The novelty of the solution is both in the LP itself, and in the rounding, which is based on a delicate decomposition of the instance into  $O(\log \log m)$  many sub-instances and on showing that, loosely speaking, the load due to each sub-instance is at most  $O(\alpha\lambda)$ . By binary-searching on the value of the optimal makespan, and rescaling, we can assume that the optimal makespan is between  $\frac{1}{2}$  and 1.

**The LP Relaxation.** Consider an instance  $\mathcal{I}$  of GENMAKESPAN given by a set of  $n$  tasks and  $m$  resources, with sets  $U_j$  and  $L_i$  as described in Sect. 2. We give an LP relaxation which is feasible if the optimal makespan is at most one.

**Lemma 5.** Consider any feasible solution to  $\mathcal{I}$  that selects a subset  $S \subseteq [n]$  of tasks. If the expected maximum load  $\mathbb{E} \left[ \max_{i=1}^m \sum_{j \in L_i \cap S} X_j \right] \leq 1$ , then

$$\sum_{j \in S} \mathbb{E}[X_j''] \leq 2, \quad \text{and} \quad (4)$$

$$\sum_{j \in L(K) \cap S} \beta_k(X_j') \leq b \cdot k, \quad \text{for all } K \subseteq [m], \quad \text{where } k = |K|, \quad (5)$$

for  $b$  being a large enough but fixed constant.

Lemma 5 allows us to write the following feasibility linear programming relaxation for GENMAKESPAN (assuming the optimal value is 1). For every task  $j$ , we have a binary variable  $y_j$ , which is meant to be 1 if  $j$  is selected in the solution. Moreover, we can drop all tasks  $j$  with  $c_j = \mathbb{E}[X_j''] > 2$  as such a task would never be part of an optimal solution- by (4). So in the rest of this paper we will assume that  $\max_{j \in [n]} c_j \leq 2$ . Further, note that we only use effective sizes  $\beta_k$  of truncated r.v.s, so we have  $0 \leq \beta_k(X_j') \leq 1$  for all  $k \in [m]$  and  $j \in [n]$ .

$$\sum_{j=1}^n y_j \geq t \quad (6)$$

$$\sum_{j=1}^n \mathbb{E}[X_j''] \cdot y_j \leq 2 \quad (7)$$

$$\sum_{j \in L(K)} \beta_k(X_j') \cdot y_j \leq b \cdot k \quad \forall K \subseteq [m] \text{ with } |K| = k, \forall k = 1, 2, \dots, m, \quad (8)$$

$$0 \leq y_j \leq 1 \quad \forall j \in [n]. \quad (9)$$

In the above LP,  $b \geq 1$  denotes the universal constant multiplying  $k$  in the right-hand-side of (5). This can be solved approximately in polynomial time.

**Theorem 3 (Solving the LP).** *There is a polynomial time algorithm which given an instance  $\mathcal{I}$  of GENMAKESPAN outputs one of the following:*

- a solution  $y \in \mathbb{R}^n$  to LP (6)–(9), except that the RHS of (8) is replaced by  $\frac{e}{e-1}bk$ , or
- a certificate that LP (6)–(9) is infeasible.

The  $\frac{e}{e-1}$  factor comes from checking feasibility via an approximation algorithm for the maximum coverage problem. In the rest of this section, we assume we have a feasible solution  $y$  to (6)–(9) since the approximate feasibility of (8) only affects the approximation ratio by a constant.

**Rounding.** We first give some intuition about the rounding algorithm. It involves formulating  $O(\log \log m)$  many almost-disjoint instances of the deterministic reward-maximization problem (2) used in the definition of  $\alpha$ -packability. The key aspect of each such deterministic instance is the definition of the sizes  $s_j$ : for the  $\ell^{\text{th}}$  instance we use effective sizes  $\beta_k(X_j')$  with parameter  $k = 2^{2^\ell}$ . We use the  $\lambda$ -safety property to construct these deterministic instances and the  $\alpha$ -packable property to solve them. Finally, we show that the expected makespan induced by the selected tasks is at most  $O(\alpha\beta)$  factor away from each such deterministic instance, which leads to an overall  $O(\alpha\beta \log \log m)$  ratio.

Before delving into the details, let us formulate a generalization of the reward-maximization problem mentioned in (2), which we call the DETCOST problem. An instance  $\mathcal{I}$  of the DETCOST problem consists of a set system  $([n], \mathcal{S})$ , with a size  $s_j$  and cost  $c_j$  for each element  $j \in [n]$ . It also has parameters  $\theta \geq \max_j s_j$  and  $\psi \geq \max_j c_j$ . The goal is to find a maximum cardinality subset  $V$  of  $[n]$  such that each set in  $\mathcal{S}$  is “loaded” to at most  $\theta$ , and the total cost of  $V$  is at most  $\psi$ . The DETCOST problem has the following LP relaxation:

$$\max \left\{ \sum_{j \in [n]} y_j : \sum_{j \in S} s_j \cdot y_j \leq \theta, \forall S \in \mathcal{S}; \sum_{j \in [n]} c_j \cdot y_j \leq \psi; 0 \leq y_j \leq 1 \forall j \in [n] \right\}. \quad (10)$$



**Theorem 4.** *If a set system satisfies the  $\alpha$ -packable property, there is an  $O(\alpha)$ -approximation algorithm for DETCOST relative to the LP relaxation (10).*

We now give the rounding algorithm for the GENMAKESPAN problem. The procedure is described formally in Algorithm 1. The algorithm proceeds in  $\log \log m$  iterations of the **for** loop in lines 3–7, since the parameter  $k$  is squared in line 3 for each iteration. In line 5, we identify resources  $i$  which are fractionally loaded to more than  $2b$ , where the load is measured in terms of  $\beta_{k^2}(X'_j)$  values. The set of such resources is grouped in the set  $D_\ell$ , and we define  $J_\ell$  to be the tasks which can load these resources. Ideally, we would like to remove these resources and tasks, and iterate on the remaining tasks and resources. However, the problem is that tasks in  $J_\ell$  also load resources other than  $D_\ell$ , and so  $(D_\ell, J_\ell)$  is not independent of the rest of the instance. This is where we use the  $\lambda$ -safe property: we expand  $D_\ell$  to a larger set of resources  $M_\ell$ , which will be used to show that the effect of  $J_\ell$  on resources outside  $D_\ell$  will not be significant.

---

**Algorithm 1:** Rounding Algorithm

---

**Input** : A fractional solution  $y$  to (6)–(9)

**Output:** A subset of tasks.

- 1 Initialize remaining tasks  $J \leftarrow [n]$ ;
- 2 **for**  $\ell = 0, 1, \dots, \log \log m$  **do**
- 3 Set  $k \leftarrow 2^{2^\ell}$ ;
- 4 Initialize class- $\ell$  resources  $D_\ell \leftarrow \emptyset$ ;
- 5 **while** *there is a resource*  $i \in [m] : \sum_{j \in L_i \cap J} \beta_{k^2}(X'_j) \cdot y_j > 2b$  **do**
- 6 update  $D_\ell \leftarrow D_\ell \cup \{i\}$ ;
- 7 Set  $\widetilde{L}_i \leftarrow J \cap L_i$  and  $J \leftarrow J \setminus \widetilde{L}_i$ ;
- 8 Define the class- $\ell$  tasks  $J_\ell \leftarrow \bigcup_{i \in D_\ell} \widetilde{L}_i$ ;
- 9 Use  $\lambda$ -safety on the set system  $(J_\ell, \{L_i \cap J_\ell\}_{i \in [m]})$  to get  
 $M_\ell := \text{Extend}(D_\ell)$ ;
- 10  $\rho \leftarrow 1 + \log \log m$ ;
- 11 Define class- $\rho$  tasks  $J_\rho = J$  and class- $\rho$  resources  $M_\rho := D_\rho = [m] \setminus (\bigcup_{\ell=0}^{\rho-1} D_\ell)$ ;
- 12 Define an instance  $\mathcal{C}$  of DETCOST as follows: the set system is the disjoint union of the set systems  $(J_\ell, M_\ell)$  for  $\ell = 0, \dots, \rho$ . The other parameters are:

$$\text{Sizes } s_j = \beta_{2^{2^\ell}}(X'_j) \text{ for each } j \in J_\ell, \forall 0 \leq \ell \leq \rho, \text{ bound } \theta = 2\bar{\alpha}b,$$

$$\text{Costs } c_j = \mathbb{E}[X'_j] \text{ for each } j \in [n], \text{ bound } \psi = 2\bar{\alpha},$$

where  $\bar{\alpha}$  is the approximation ratio from Theorem 4 ;

- 13 Let  $N_H = \{j \in [n] : y_j > 1/\bar{\alpha}\}$  ;
  - 14 Let  $\bar{y}_j = \bar{\alpha} \cdot y_j$  for  $j \in [n] \setminus N_H$  and  $\bar{y}_j = 0$  otherwise ;
  - 15 Round  $\bar{y}$  (as a feasible solution to (10)) using Theorem 4 to obtain  $N_L$ ;
  - 16 Output  $N_H \cup N_L$ .
- 

Consider any iteration  $\ell$  of the for-loop. We apply the  $\lambda$ -safety property to the set-system  $(J_\ell, \{L_i \cap J_\ell\}_{i \in [m]})$  and set  $D_\ell$  to get  $M_\ell := \text{Extend}(D_\ell)$ . We

remove  $J_\ell$  from the current set  $J$  of tasks, and continue to the next iteration. We abuse notation by referring to  $(J_\ell, M_\ell)$  as the following set system: each set is of the form  $L_i \cap J_\ell$  for some  $i \in M_\ell$ . Having partitioned the tasks into classes  $J_1, \dots, J_\rho$ , we consider the disjoint union  $\mathcal{D}$  of the set systems  $(J_\ell, M_\ell)$ , for  $\ell = 1, \dots, \rho$ . While the sets  $D_\ell$  are disjoint, the sets  $M_\ell$  may not be disjoint. For each resource appearing in the sets  $M_\ell$  of multiple classes, we make distinct copies in the combined set-system  $\mathcal{D}$ .

Finally, we set up an instance  $\mathcal{C}$  of DETCOST (in line 11): the set system is the disjoint union of  $(J_\ell, M_\ell)$ , for  $\ell = 1, \dots, \rho$ . Every task  $j \in J_\ell$  has size  $\beta_{2^{2\ell}}(X'_j)$  and cost  $\mathbb{E}[X'_j]$ . The parameters  $\theta$  and  $\psi$  are as mentioned in line 11. Our proofs show that the solution  $\bar{y}$  defined in line 14 is a feasible solution to the LP relaxation (10) for  $\mathcal{C}$ . This allows us to use Theorem 4 to round  $\bar{y}$  to an integral solution  $N_L$ . Finally, we output  $N_H \cup N_L$ , where  $N_H$  is defined in line 12.

The analysis is outlined in the appendix.

## 4 Applications

As discussed in Sect. 2, the problem of selecting intervals in a line satisfies the  $\lambda$ -safe property with  $\lambda = 2$ . Moreover, the  $\alpha$ -packable property holds with  $\alpha = O(1)$ , which follows from [3]—indeed, the LP relaxation (2) corresponds to the unsplittable flow problem where all vertices have uniform capacity  $\theta$ . So, Theorem 2 now implies the following.

**Corollary 1.** *There is an  $O(\log \log m)$ -approximation for GENMAKESPAN where resources are vertices on a line and tasks are intervals in this line.*

The full version [11] has a number of other applications:

**Corollary 2.** *There is an  $O(\log \log m)$ -approximation for GENMAKESPAN when*

- *resources are vertices in a tree and tasks are paths in this tree.*
- *resources are all points in the plane and tasks are rectangles, where the rectangles in a solution can be shrunk by a  $(1-\delta)$ -factor in either dimension;  $\delta > 0$  is some constant.*
- *resources are all points in the plane and tasks are disks.*

## 5 Integrality Gap Lower Bounds

We consider two natural questions – (i) does one require any assumption on the underlying set system to obtain  $O(1)$ -approximation for GENMAKESPAN?, and (ii) what is the integrality gap of the LP relaxation given by the constraints (6)–(9) for settings where  $\alpha$  and  $\lambda$  are constants? For the first question, we show that applying our LP based approach to general set systems only gives an  $\Omega\left(\frac{\log m}{(\log \log m)^2}\right)$  approximation ratio, and so we do require some conditions on the

underlying set system. For the second question, we show that even for set systems given by intervals on a line (as in Sect. 4), the integrality gap of our LP relaxation is  $\Omega(\log^* m)$ . Hence this rules out getting a constant-factor approximation using our approach even when  $\alpha$  and  $\lambda$  are constants.

## A Analysis Outline

We now show that the expected makespan for the solution produced by the rounding algorithm above is  $O(\alpha\lambda\rho)$ , where  $\rho = \log \log m$  is the number of classes. In particular, we show that the expected makespan (taken over all resources) due to tasks of each class  $\ell$  is  $O(\alpha\lambda)$ .

Using the terminating condition in line 5, we can show:

**Lemma 6.** *For any class  $\ell$ ,  $0 \leq \ell \leq \rho$ , and resource  $i \in [m]$ ,*

$$\sum_{j \in J_\ell \cap L_i} \beta_r(X'_j) \cdot y_j \leq 2b, \quad \text{where } r = 2^{2^\ell}.$$

Next, the sets  $D_\ell$  cannot become too large (as a function of  $\ell$ ).

**Lemma 7.** *For any  $\ell$ ,  $0 \leq \ell \leq \rho$ ,  $|D_\ell| \leq k^2$ , where  $k = 2^{2^\ell}$ . So  $|M_\ell| \leq k^p$  for some constant  $p$ .*

*Proof.* The claim is trivial for the last class  $\ell = \rho$  as  $k \geq m$  in this case. Now consider any class  $\ell < \rho$ . For each  $i \in D_\ell$ , we know  $\sum_{j \in \widetilde{L}_i} \beta_{k^2}(X'_j) \cdot y_j > 2b$ , where  $\widetilde{L}_i$  is as defined in line 7. Moreover, the subsets  $\{\widetilde{L}_i : i \in D_\ell\}$  are disjoint as the set  $J$  gets updated (in line 7) after adding each  $i \in D_\ell$ . Suppose, for the sake of contradiction, that  $|D_\ell| > k^2$ . Then let  $K \subseteq D_\ell$  be any set of size  $k^2$ . By the LP constraint (8) on this subset  $K$ ,

$$2b \cdot k^2 < \sum_{i \in K} \sum_{j \in \widetilde{L}_i} \beta_{k^2}(X'_j) \cdot y_j \leq \sum_{j \in L(K)} \beta_{k^2}(X'_j) \cdot y_j \leq b|K| = b \cdot k^2,$$

which is a contradiction. This proves the first part of the claim. Finally, the  $\lambda$ -safe property implies that  $|M_\ell|$  is polynomially bounded by  $|D_\ell|$ . ■

Using the definition of the DETCOST instance and Lemma 6, we can show:

**Lemma 8.** *The fractional solution  $\bar{y}$  is feasible for the LP relaxation (10) corresponding to the DETCOST instance  $\mathcal{C}$ . Moreover,  $\theta \geq \max_j s_j$  and  $\psi \geq \max_j c_j$ .*

The above lemmas show that the algorithm is well-defined, so we can indeed use Theorem 4 to round  $\bar{y}$  into an integer solution. Recall that our final solution is  $N = N_H \cup N_L$ . The next two lemmas follow from this rounding step.

**Lemma 9.**  $|N_H| + |N_L| \geq t$ .

**Lemma 10.** For any class  $\ell \leq \rho$  and resource  $i \in M_\ell$ ,

$$\sum_{j \in N_\ell \cap L_i} \beta_k(X'_j) \leq 4\bar{\alpha}b, \quad \text{where } k = 2^{2^\ell}.$$

We now focus on a particular class  $\ell \leq \rho$  and show that the expected makespan due to tasks in  $N \cap J_\ell$  is small. Recall that  $k = 2^{2^\ell}$ . Let  $N_\ell := N \cap J_\ell$  and let  $\text{Load}_i^{(\ell)} := \sum_{j \in N_\ell \cap L_i} X'_j$  be the load on any resource  $i \in [m]$  due to class- $\ell$  tasks. We can now bound the makespan due to the truncated random variables.

**Lemma 11.** For any class  $\ell \leq \rho$ ,  $\mathbb{E} \left[ \max_{i \in M_\ell} \text{Load}_i^{(\ell)} \right] \leq 4\bar{\alpha}b + O(1)$ , and

$$\mathbb{E} \left[ \max_{i=1}^m \text{Load}_i^{(\ell)} \right] \leq 4\lambda\bar{\alpha}b + O(\lambda) = O(\alpha\lambda).$$

*Proof.* Consider a resource  $i \in M_\ell$ . Lemmas 10 and 3 imply:

$$\Pr \left[ \text{Load}_i^{(\ell)} > 4\bar{\alpha}b + \gamma \right] = \Pr \left[ \sum_{j \in N_\ell \cap L_i} X'_j > 4\bar{\alpha}b + \gamma \right] \leq k^{-\gamma}, \quad \forall \gamma \geq 0.$$

By a union bound, we get

$$\Pr \left[ \max_{i \in M_\ell} \text{Load}_i^{(\ell)} > 4\bar{\alpha}b + \gamma \right] \leq |M_\ell| \cdot k^{-\gamma} \leq k^{p-\gamma}, \quad \text{for all } \gamma \geq 0,$$

where  $p$  is the constant from Lemma 7. So the expectation

$$\begin{aligned} \mathbb{E} \left[ \max_{i \in M_\ell} \text{Load}_i^{(\ell)} \right] &= \int_{\theta=0}^{\infty} \Pr \left[ \max_{i \in M_\ell} \text{Load}_i^{(\ell)} > \theta \right] d\theta \\ &\leq 4\bar{\alpha}b + p + 2 + \int_{\gamma=p+2}^{\infty} k^{-\gamma+p} d\gamma \leq 4\bar{\alpha}b + p + 2 + \frac{1}{k(p+1)}, \end{aligned}$$

which completes the proof of the first statement.

We now prove the second statement. Consider any class  $\ell < \rho$ : by definition of  $J_\ell$ , we know that  $J_\ell \subseteq L(D_\ell)$ . So the  $\lambda$ -safe property implies that for every resource  $i$  there is a subset  $R_i \subseteq M_\ell$  of size at most  $\lambda$  such that  $L_i \cap J_\ell \subseteq L(R_i) \cap J_\ell$ . Because  $N_\ell \subseteq J_\ell$ , we also have  $L_i \cap N_\ell \subseteq L(R_i) \cap N_\ell$ . Therefore,

$$\text{Load}_i^{(\ell)} \leq \sum_{z \in R_i} \text{Load}_z^{(\ell)} \leq \lambda \max_{z \in M_\ell} \text{Load}_z^{(\ell)}.$$

Taking expectation on both sides, we obtain the desired result.

Finally, for the last class  $\ell = \rho$ , note that any task in  $J_\rho$  loads the resources in  $D_\rho = M_\rho$  only. Therefore,  $\max_{i=1}^m \text{Load}_i^{(\ell)} = \max_{z \in M_\ell} \text{Load}_z^{(\ell)}$ . The desired result now follows by taking expectation on both sides.  $\blacksquare$

Using Lemma 11 for all  $\rho$  classes, it follows that the expected makespan due to all truncated r.v.s is  $O(\alpha\lambda\rho)$ . For the exceptional random variables, we use:

**Lemma 12.**  $\mathbb{E} \left[ \sum_{j \in N} X_j'' \right] = \sum_{j \in N} c_j \leq 4\bar{\alpha}$ .

Adding the contributions from truncated and exceptional r.v.s, the overall expected makespan is  $O(\alpha\lambda\rho)$ , which completes the proof of Theorem 2.

## References

1. Adamaszek, A., Chalermsook, P., Wiese, A.: How to tame rectangles: solving independent set and coloring of rectangles via shrinking. In: APPROX/RANDOM, pp. 43–60 (2015)
2. Agarwal, P.K., Mustafa, N.H.: Independent set of intersection graphs of convex objects in 2D. *Comput. Geom.* **34**(2), 83–95 (2006)
3. Chakrabarti, A., Chekuri, C., Gupta, A., Kumar, A.: Approximation algorithms for the unsplittable flow problem. *Algorithmica* **47**(1), 53–78 (2007)
4. Chalermsook, P.: Coloring and maximum independent set of rectangles. In: Goldberg, L.A., Jansen, K., Ravi, R., Rolim, J.D.P. (eds.) APPROX/RANDOM 2011. LNCS, vol. 6845, pp. 123–134. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-22935-0\\_11](https://doi.org/10.1007/978-3-642-22935-0_11)
5. Chalermsook, P., Chuzhoy, J.: Maximum independent set of rectangles. In: SODA, pp. 892–901 (2009)
6. Chan, T.M.: A note on maximum independent sets in rectangle intersection graphs. *Inf. Process. Lett.* **89**(1), 19–23 (2004)
7. Chan, T.M., Har-Peled, S.: Approximation algorithms for maximum independent set of pseudo-disks. *Discret. Comput. Geom.* **48**(2), 373–392 (2012)
8. Chekuri, C., Mydlarz, M., Shepherd, F.B.: Multicommodity demand flow in a tree and packing integer programs. *ACM Trans. Algorithms* **3**(3), 27 (2007)
9. Elwalid, A.I., Mitra, D.: Effective bandwidth of general markovian traffic sources and admission control of high speed networks. *IEEE/ACM Trans. Netw.* **1**(3), 329–343 (1993)
10. Gupta, A., Kumar, A., Nagarajan, V., Shen, X.: Stochastic load balancing on unrelated machines. In: SODA, pp. 1274–1285. Society for Industrial and Applied Mathematics (2018)
11. Gupta, A., Kumar, A., Nagarajan, V., Shen, X.: Stochastic makespan minimization in structured set systems. *arXiv* (2020). <https://arxiv.org/abs/2002.11153>
12. Hui, J.Y.: Resource allocation for broadband networks. *IEEE J. Sel. Areas Commun.* **6**(3), 1598–1608 (1988)
13. Kelly, F.P.: Notes on effective bandwidths. In: *Stochastic Networks: Theory and Applications*, pp. 141–168. Oxford University Press (1996)
14. Kleinberg, J., Rabani, Y., Tardos, E.: Allocating bandwidth for bursty connections. *SIAM J. Comput.* **30**(1), 191–217 (2000)
15. Molinaro, M.: Stochastic  $\ell_p$  load balancing and moment problems via the l-function method. In: SODA, pp. 343–354 (2019)