Stochastic Makespan Minimization in Structured Set Systems

Anupam Gupta[*] Amit Kumar[†] Viswanath Nagarajan[‡] Xiangkun Shen[§]

December 9, 2019

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. Jobs 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 that this LP has an $\Omega(\log^* m)$ integrality gap even for the problem of selecting intervals on a line. Moreover, we show logarithmic gaps for problems without geometric structure, showing that some structure is needed to get good results using these techniques.

[*]Computer Science Department, Carnegie Mellon University, Pittsburgh, USA. Supported in part by NSF award CCF-1907820, and the Indo-US Joint Center for Algorithms Under Uncertainty.

[†]Dept. of Computer Science and Engg., IIT Delhi, India 110016.

[‡]Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109. Supported in part by NSF CAREER grant CCF-1750127.

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

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

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 [4].

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. [7, 2, 9, 8, 6, 1, 4].

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_js 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 α-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 λ resources in D.

These assumptions are formalized in §2. To give some intuition for these assumptions, consider intervals on the line. The first assumption holds by the results of [4]. The second assumption holds because each resource is some time τ, and the tasks using time τ can be covered by two resources in D, namely times $\tau_1, \tau_2 \in D$ such that $\tau_1 \leq \tau \leq \tau_2$.

Our informal main result is the following:

Theorem 1.1 (Main (Informal))**.** *There is an $O(\alpha\lambda \log\log m)$-approximation algorithm for stochastic makespan minimization (*GenMakespan*), with α and λ as in the above assumptions.*

We also show that both α and λ are constant in a number of geometric settings: for intervals on a line, for paths in a tree, and for rectangles and "fat objects" 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 α-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 [14, 11]. In the example above, this would handle cases 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 [4], paths in a tree (with edge loads) [9] and rectangles in a plane (under some restrictions) [1].

Our techniques draw on prior work on stochastic makespan minimization for identical [14] and unrelated [11] 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 [11]. The "volume" constraints in our LP also has some similarity to those in [11]: however, a key difference here is that the random variables loading different resources are correlated (whereas they were independent in [11]). Indeed, this is why our LP can only be solved approximately whereas the LP relaxation in [11] was optimally solvable. We emphasize that our main contribution is the rounding algorithm ideas uses a new set of ideas; these lead to the $O(\log\log m)$ approximation bound, whereas the rounding in [11] 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 ℓ_p-norms (note that the makespan corresponds to the ℓ_∞-norm) and a constant-factor approximation is known [15]. We do not consider ℓ_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*:

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

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 $i \in [m]$. Note that the set system has n elements (corresponding to tasks) and m sets (corresponding to resources).

A1 **(α-packable)**: A set system $([n], \mathcal{L})$ is said to be *α-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 $\widehat{y}$, losing a factor of at most $\alpha \geq 1$. (I.e., $\sum_j r_j \widehat{y}_j \geq \frac{1}{\alpha} \sum_j r_j y_j$.)

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

We also assume that the support of $\widehat{y}$ is contained in the support of y.[1]

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

A2 (λ-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 λ-*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 $i \in [m]$, 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 $\texttt{Extend}(D)$.

Let us give an example. Suppose $P = [m]$ are m points on the line, and consider n intervals $I_1, \ldots, I_n$ of the line with each $I_j \subseteq P$. Now the set system is defined on n elements (one for each interval), with m sets with the set L_i for point $i \in [m]$ containing the indices of intervals that contain i. The λ-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 on the line, there are λ 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 λ 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 α-packable and λ-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 fat-objects in a plane, and each L_i consists of the elements 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 $\mathcal{L}|_X = \{L \cap X \mid L \in \mathcal{L}\}$. Loosely speaking, the following lemma (proved in Appendix A) formalizes that packability and safeness properties also hold for sub-families and disjoint unions.

Lemma 2.1. *Consider a set system $([n], \mathcal{L})$ that is α-packable and λ-safe. Then,*

(i) for all $X \subseteq [n]$, the set system $(X, \mathcal{L})$ is α-packable and λ-safe, and

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

We consider the GenMakespan problem for settings where the set system $([n], \{L_i\}_{i \in [m]})$ is α-packable and λ-safe for some small parameters α and λ. We show in §4 that the families discussed above satisfy these properties. Our main result is the following:

Theorem 2.2. *For any instance of* GenMakespan *where the corresponding set system $([n], \{L_i\}_{i \in [m]})$ is α-packable and λ-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). For each $j \in [n]$ we define $c_j := \mathbb{E}[X''_j]$ to be the expected exceptional size. The following result was shown in [14]:

Lemma 2.3 (Exceptional Items Lower Bound)**.** *Let $X''_1, X''_2, \ldots, X''_t$ 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) [12, 13, 10, 11].

Definition 2.4 (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 $Y_1, \ldots, Y_k$ 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 2.5 (Effective Size: Upper Bound)**.** *For indep. r.v.s $Y_1, \ldots, Y_n$, 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 2.6 (Effective Size: Lower Bound)**.** *Let $X_1, X_2, \cdots X_n$ 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 prove Theorem 2.2: given a set system that is α-packable and λ-safe, we show an $O(\alpha\lambda \log\log m)$-approximation algorithm. 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 $O(\alpha\lambda)$.

3.1 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 §2. 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. We give an LP relaxation which is feasible if the optimal makespan is at most one. We use properties of truncated and exceptional random variables; recall the definitions of these r.v.s from §2.2.

Lemma 3.1. *Consider any feasible solution to $\mathcal{I}$ that selects a subset $S \subseteq [n]$ of tasks. If the expected maximum load $\mathbb{E}\left[\max_{i=1}^m \sum_{j\in L_i \cap S} X_j\right] \le 1$, then*

$$\sum_{j\in S} \mathbb{E}[X''_j] \le 2, \qquad \textit{and} \tag{4}$$

$$\sum_{j\in L(K)\cap S} \beta_k(X'_j) \;\le\; b\cdot k, \; \textit{for all } K \subseteq [m], \quad \textit{where } k = |K|, \tag{5}$$

for b being a large enough but fixed constant.

Proof. The first inequality (4) follows follows from Lemma 2.3 applied to $\{X''_j : j \in S\}$ and $L = 1$.

For the second inequality (5), consider any subset $K \subseteq [m]$ of the resources. Let $\widetilde{L}_i \subseteq L_i$ for $i \in K$ be such that $\{\widetilde{L}_i\}_{i\in K}$ forms a partition of $\bigcup_{i\in K}(L_i \cap S) = L(K) \cap S$. Then, we apply Lemma 2.6 to the resources in K and the truncated random variables $\{X'_j : j \in \bigcup_{i\in K} \widetilde{L}_i\}$. Because, $\mathbb{E}[\max_{i\in K} \sum_{j\in\widetilde{L}_i} X'_j] \le \mathbb{E}[\max_{i\in K} \sum_{j\in L_i\cap S} X'_j] \le 1$, the contrapositive of Lemma 2.6 implies $\sum_{j\in\cup_{i\in K}\widetilde{L}_i} \beta_k(X'_j) \le b\cdot k$, where $b = O(1)$ is a fixed constant. Inequality (5) now follows from the fact that $\cup_{i\in K}\widetilde{L}_i = L(K)\cap S$. □

Lemma 3.1 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 $\max_{j\in[n]} c_j \le 2$. Further, note that we only use effective sizes β_k of *truncated* r.v.s, so we have $0 \le \beta_k(X'_j) \le 1$ for all $k \in [m]$ and $j \in [n]$.

$$\sum_{j=1}^n y_j \ge t \tag{6}$$

$$\sum_{j=1}^n \mathbb{E}[X''_j]\cdot y_j \le 2 \tag{7}$$

$$\sum_{j\in L(K)} \beta_k(X'_j)\cdot y_j \le b\cdot k \qquad \forall K \subseteq [m] \text{ with } |K| = k,\ \forall k = 1,2,\cdots m, \tag{8}$$

$$0 \le y_j \le 1 \qquad \forall j \in [n]. \tag{9}$$

In the above LP, $b \ge 1$ denotes the universal constant multiplying k in the right-hand-side of (5). In Appendix B we show that it can be solved approximately in polynomial time.

Theorem 3.2 (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.*

In the rest of this section, we assume we have a feasible solution y to (6)–(9), and ignore the fact that we only satisfy (8) up to a factor of $\frac{e}{e-1}$, since it affects the approximation ratio by a constant.

3.2 Rounding a Feasible LP Solution

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 α-packability. The key aspect of each such deterministic instance is the definition of the sizes s_j: for the ℓ^{th} instance we use effective sizes $\beta_k(X'_j)$ with parameter $k = 2^{2^\ell}$. We use the λ-safety property to construct these deterministic instances and the α-packable property to solve them. Finally, we show that the expected makespan induced by the selected tasks is at most $O(\alpha\lambda)$ factor away from each such deterministic instance, which leads to an overall $O(\alpha\lambda\log\log m)$-approximation 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 θ, and the total cost of V is at most ψ. The DetCost problem has the following LP relaxation:

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

The following result, whose proof is deferred to the appendix, shows that the α-packable property for a set system implies an $O(\alpha)$-approximation for the DetCost problem for it.

Theorem 3.3. *Suppose a set system satisfies the α-packable property. Then 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 $\log\log m$ rounds 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_ℓ, and we define J_ℓ 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 jobs in J_ℓ also load resources other than D_ℓ, and so (D_ℓ, J_ℓ) is not independent of the rest of the instance. This is where we use the λ-safe property: we expand D_ℓ to a larger set of resources M_ℓ, which will be used to show that the effect of J_ℓ on resources outside D_ℓ will not be significant.

If J denotes the current sets of tasks, let J_ℓ be the set of current tasks which load a resource in D_ℓ (defined in line 7). We apply the λ-safety property to the set-system $(J_\ell, \{L_i \cap J_\ell\}_{i\in[m]})$ and set D_ℓ to get $M_\ell := \texttt{Extend}(J_\ell)$. We remove J_ℓ from J, and continue to the next iteration. We abuse notation by referring to (J_ℓ, M_ℓ) as the following set system: each set in this set system is of the form $L_i \cap J_\ell$ for some $i \in M_\ell$. Having partitioned the tasks into classes $J_1, \ldots, J_\rho$, we consider the disjoint union $\mathcal{D}$ of the set systems (J_ℓ, M_ℓ), for $\ell = 1, \ldots, \rho$. While the set D_ℓ are disjoint, the sets M_ℓ may not be disjoint. For each resource appearing in the sets M_ℓ 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_ℓ, M_ℓ), for $\ell = 1, \ldots, \rho$. Every task $j \in J_\ell$ has size $\beta_{2^{2^\ell}}(X'_j)$ and cost $\mathbb{E}[X''_j]$. The parameters θ and ψ 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 3.3 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.

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 $\ell = 0, 1, \ldots, \log\log m$ 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 $M_\ell :=$ Extend$(D_\ell)$ ;
10 $\rho \leftarrow 1 + \log\log m$;
11 Define class-$\rho$ tasks $J_\rho = J$ and class-$\rho$ resources $M_\rho := D_\rho = [m] \setminus \left(\cup_{\ell=0}^{\rho-1} D_\ell\right)$ ;
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 $\ell = 0, \ldots, \rho$. The other parameters are as follows:

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

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

     where $\bar{\alpha}$ is the approximation ratio from Theorem 3.3 ;
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 3.3 to obtain $N_L$;
16 Output $N_H \cup N_L$.
```

3.3 The Analysis

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, the expected makespan (taken over all the resources) due to tasks of each class ℓ is $O(\alpha\lambda)$.

Claim 3.4. *For any class ℓ, $0 \le \ell \le \rho$, and resource $i \in [m]$,*

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

Proof. If $\ell = 0$, we apply the LP constraint (8) for a subset $\{i, i'\}$ of size two containing the resource i to get:

$$\sum_{j \in L_i} \beta_2(X'_j) \cdot y_j \le \sum_{j \in L(\{i,i'\})} \beta_2(X'_j) \cdot y_j \le 2b,$$

which implies the desired result.

So assume $\ell \geq 1$. Let J denote the set of remaining tasks at the end of iteration $\ell - 1$, i.e., $J = \cup_{\ell' \geq \ell} J_{\ell'}$. The terminating condition in line 5 implies that

$$\sum_{j \in J \cap L_i} \beta_r(X'_j) \cdot y_j \leq 2b, \text{ for all } i \in [m],$$

which implies the lemma. □

Next, the sets D_ℓ and M_ℓ cannot become too large (as a function of ℓ).

Claim 3.5. *For any ℓ, $0 \leq \ell \leq \rho$, $|D_\ell| \leq k^2$, 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 λ-safe property implies that $|M_\ell|$ is polynomially bounded by $|D_\ell|$. □

We now show that $\bar{y}$ from line 14 is feasible to the LP relaxation for DETCOST given in (10).

Claim 3.6. *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 $\psi \geq \max_j c_j$.*

Proof. Note that $0 \leq \bar{y} \leq 1$ by construction. Since the sets J_ℓ partition $[n]$,

$$\sum_{\ell=0}^{\rho} \sum_{j \in J_\ell} c_j \cdot \bar{y}_j = \sum_{j \in [n]} c_j \cdot \bar{y}_j \leq \bar{\alpha} \sum_j c_j \cdot y_j \leq 2\bar{\alpha} = \psi$$

where the last inequality follows from the feasibility of constraint (7).

To verify the size constraint for each resource i in the disjoint union of M_ℓ for $\ell = 0, \ldots, \rho$, consider any such class ℓ and $i \in M_\ell$. The size constraint for i is:

$$\sum_{j \in J_\ell \cap L_i} \beta_k(X'_j) \cdot \bar{y}_j \leq \theta = 2\bar{\alpha}b, \tag{11}$$

where $k = 2^{2^\ell}$. Since $\bar{y} \leq \bar{\alpha} \cdot y$, this follows directly from Claim 3.4.

Finally, since the truncated sizes X'_j lie in $[0, 1]$, so do their effective sizes. Hence $s_j \in [0, 1] \leq \theta$. Moreover, as assumed earlier, by (4) we have $c_j = \mathbb{E}[X''_j] \leq 2 \leq \psi$ for all $j \in [n]$. □

The above claims show that the algorithm is well-defined, so we can use Theorem 3.3 to round $\bar{y}$ into an integer solution. The next claims show properties of this solution. First, the set of tasks output by Algorithm 1 has the desired cardinality.

Claim 3.7. $|N_H| + |N_L| \geq t$.

Proof. Note that by the feasibility of the constraint (6), $\sum_{j\in[n]\setminus N_H} y_j \geq t - |N_H|$. Further, $\bar{y}_j = \bar{\alpha}\cdot y_j \in [0,1]$ for all tasks $j \in [n]\setminus N_H$. Therefore,

$$|N_L| \geq \frac{1}{\bar{\alpha}} \sum_{j\in[n]\setminus N_H} \bar{y}_j = \sum_{j\in[n]\setminus N_H} y_j \geq t - |N_H|,$$

which completes the proof. □

Our final solution is $N = N_H \cup N_L$. 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 $k = 2^{2^\ell}$. For sake of brevity, let $N_\ell := N \cap J_\ell$ and let $\mathsf{Load}_i^{(\ell)} := \sum_{j\in N_\ell\cap L_i} X'_j$ denote the load on any resource $i \in [m]$ due to class-ℓ tasks. The following claim is the "rounded" version of Claim 3.4.

Claim 3.8. *For any class $\ell \leq \rho$ and resource $i \in M_\ell$,*

$$\sum_{j\in N_\ell\cap L_i} \beta_k(X'_j) \leq 4\bar{\alpha}b, \quad \text{where } k = 2^{2^\ell}. \tag{12}$$

Proof. Since $N_\ell \cap L_i = (N_H \cap J_\ell \cap L_i) \cup (N_L \cap J_\ell \cap L_i)$, we bound the LHS above in two parts. By Claim 3.4, the solution y satisfies $\sum_{j\in J_\ell\cap L_i} \beta_k(X'_j)\cdot y_j \leq 2b$. As each task $j \in N_H$ has $y_j > 1/\bar{\alpha}$,

$$\sum_{j\in N_H\cap J_\ell\cap L_i} \beta_k(X'_j) \leq 2\bar{\alpha}b.$$

Since N_L is a feasible integral solution to (10), the size constraint for $i \in M_\ell$ implies that

$$\sum_{j\in N_L\cap J_\ell\cap L_i} \beta_k(X'_j) = \sum_{j\in N_L\cap J_\ell\cap L_i} s_j \leq \theta = 2\bar{\alpha}b.$$

Combining the two bounds above, we obtain the claim. □

We are now ready to bound the makespan due to the truncated part of the random variables.

Lemma 3.9. *Consider any class $\ell \leq \rho$. We have $\mathbb{E}\left[\max_{i\in M_\ell} \mathsf{Load}_i^{(\ell)}\right] \leq 4\bar{\alpha}b + O(1)$. Therefore,*

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

Proof. Consider a resource $i \in M_\ell$. Claim 3.8 and Lemma 2.5 imply that for any $\gamma > 0$,

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

By a union bound, we get

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

where p is the constant from Claim 3.5. So the expectation

$$\mathbb{E}\left[\max_{i\in M_\ell} \mathsf{Load}_i^{(\ell)}\right] = \int_{\theta=0}^{\infty} \Pr\left[\max_{i\in M_\ell} \mathsf{Load}_i^{(\ell)} > \theta\right] d\theta$$

$$\leq \; 4\bar{\alpha}b + p + 2 + \int_{\gamma=p+2}^{\infty} k^{-\gamma+p}\, d\gamma \;\leq\; 4\bar{\alpha}b + p + 2 + \frac{1}{k(p+1)},$$

which completes the proof of the first statement.

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

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

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

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

Using Lemma 3.9, we can bound the expected makespan due to all truncated random variables:

$$\mathbb{E}\left[\max_{i=1}^{m} \sum_{j \in N \cap L_i} X'_j\right] = \mathbb{E}\left[\max_{i=1}^{m} \sum_{\ell=0}^{\rho} \mathsf{Load}_i^{(\ell)}\right] \leq \sum_{\ell=0}^{\rho} \mathbb{E}\left[\max_{i=1}^{m} \mathsf{Load}_i^{(\ell)}\right] \leq O(\alpha\lambda\rho). \tag{13}$$

For the exceptional random variables, we use the following simple fact:

Claim 3.10. $\mathbb{E}\left[\sum_{j \in N} X''_j\right] = \sum_{j \in N} c_j \leq 4\bar{\alpha}$.

Proof. Feasibility of constraint (7) implies that $\sum_{j=1}^n c_j \cdot y_j \leq 2$. As each task $j \in N_H$ has $y_j > 1/\bar{\alpha}$, we have $\sum_{j \in N_H} c_j \leq 2\bar{\alpha}$. For tasks in N_L, the fact that N_L is a feasible integral solution to (10) implies that $\sum_{j \in N_L} c_j \leq \psi = 2\bar{\alpha}$. This completes the proof. □

Finally, using (13) and Claim 3.10, we have:

$$\mathbb{E}\left[\max_{i=1}^{m} \sum_{j \in N \cap L_i} X_j\right] = \mathbb{E}\left[\max_{i=1}^{m} \sum_{j \in N \cap L_i} (X'_j + X''_j)\right] \leq \mathbb{E}\left[\max_{i=1}^{m} \sum_{j \in N \cap L_i} X'_j\right] + \mathbb{E}\left[\sum_{j \in N} X''_j\right] \leq O(\alpha\lambda\rho).$$

This completes the proof of Theorem 2.2.

4 Applications

In this section, we show that several stochastic optimization problems of interest that have nice structural properties also satisfy the two assumptions of α-packability and λ-safety for small values of these parameters (typically $\alpha, \lambda = O(1)$ in these problems), and hence GenMakespan can be solved efficiently using our framework.

4.1 Intervals on a Line

We are given a line graph on n vertices. The resources are given by the set of vertices in this line. Each task corresponds to an interval in this line. So each task j loads all the vertices in the corresponding interval. For each vertex i, L_i denotes the set of tasks (i.e., intervals) which contain i.

The α-packable property for this set system with $\alpha = O(1)$ follows from the result in [4]—indeed, the LP relaxation (2) corresponds to the unsplittable flow problem where all vertices have uniform capacity θ. We now show the λ safe property.

Lemma 4.1. *The above set system is 2-safe.*

Proof. Consider a subset D of vertices. We define $M := \texttt{Extend}(D)$ to be same as D. For a vertex i, let l_i and r_i denote the closest vertices in M to the left and to the right of i respectively (if $i \in M$, then both these vertices are same as i). Define R_i as $\{l_i, r_i\}$. It remains to show that $L_i \cap L(D) \subseteq L(R_i) \cap L(D)$. This is easy to see. Consider a task j represented by an interval I_j which belongs to $L_i \cap L(D)$. Then I_j contains i and a vertex from D. But then it must contain either l_i or r_i. Therefore, it belongs to $L(R_i) \cap L(D)$ as well. □

Theorem 2.2 now implies the following.

Corollary 4.2. *There is an $O(\log \log m)$-approximation algorithm for* GenMakespan *where the resources are represented by vertices on a line and tasks by intervals in this line.*

4.2 Paths on a Tree

We are given a tree $T = (V, E)$ on m vertices, and a set of n paths, $\{P_j : j \in [n]\}$, in this tree. The resources correspond to vertices and tasks to paths. For a vertex $i \in [m]$, L_i is defined as the set of paths which contain i. We first show the λ-safe property for a constant λ.

Lemma 4.3. *The set system $([n], \{L_i : i \in [m]\})$ is 2-safe.*

Proof. Let D be a subset of vertices. We define $M := \texttt{Extend}(D)$ as follows: let T' be the minimal sub-tree of T which contains all the vertices in D. Note that all leaves of T' must belong to D. Then M contains D and all the vertices in T' which have degree at least three (in the tree T'). It is easy to check that $|M| \leq 2|D|$. Fix a vertex $i \in V$. We need to define R_i such that $L_i \cap L(D) \subseteq L(R_i) \cap L(D)$. Let v_i be the vertex in the sub-tree T' that has the least distance to i (if $i \in T'$, then v_i is same as i). Note that if v_i has degree 2 (in the tree T), it may not lie in M. See also Figure 4.2. We claim that:

$$L_i \cap L(D) \subseteq L_{v_i} \cap L(D) \tag{14}$$

In other words, a path P_j containing i and a vertex w in D must contain v_i as well. Indeed, the last vertex in T' (as we go from w to i) must also be the closest vertex to i in T'. We now consider two cases:

- If $v_i \in M$, we set $R_i = \{v_i\}$. By (14) we have $L_i \cap L(D) \subseteq L_{v_i} \cap L(D) = L(R_i) \cap L(D)$.
- If $v_i \notin M$ then v_i must be a degree-2 vertex in T'. Let a_i and b_i be the first two vertices of M that we encounter if we move from v_i (along the sub-tree T') in both directions. Set $R_i := \{a_i, b_i\}$. Observe that the path from a_i to b_i in T' contains v_i and all internal vertices have degree 2 (in T'), and none of them belong to M. Let P_j be a path which contains i and a vertex w in D. Since w lies in T', P_j must pass through v_i. The part of P_j from v_i to w must lie in T' and so would contain either a_i or b_i.

Since $|R_i| \leq 2$, the desired result follows. □

We now consider the α-packable property. As in the case of the line graph in the previous section, this is same as bounding the integrality gap of the unsplittable flow problem on trees where vertices have capacities. An analogous result with edge capacities was given by Chekuri et al. [9], and our rounding algorithm is inspired by their approach.

So consider an instance of the unsplittable flow problem where every vertex in the tree has capacity θ, and path P_j has reward r_j and size s_j (we assume that $\theta \geq \max_j s_j$). Our goal is to find a maximum reward subset of paths which obey the vertex capacities—we call this problem `UFP-Tree`. It is easy to see that (2) is the natural LP relaxation for this problem.

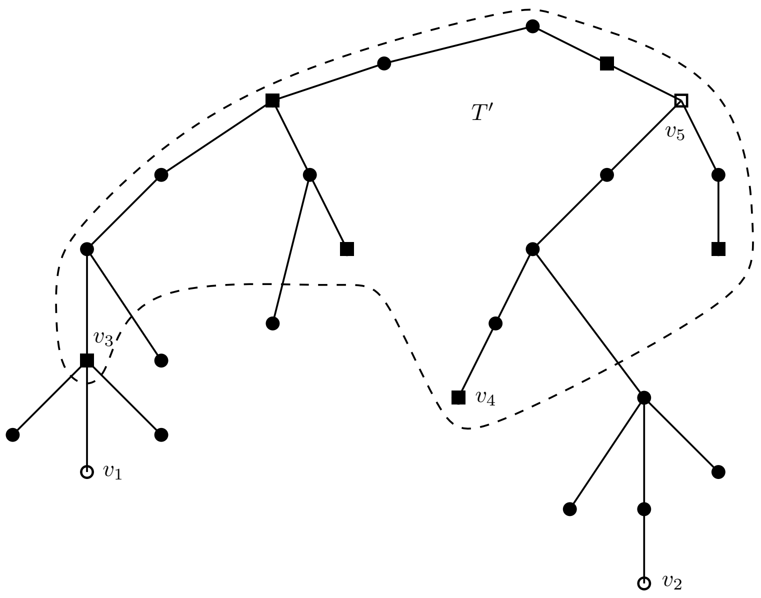

Figure 1: Tree set-system example. The solid-square vertices are the "dangerous" vertices. The box vertices are the additional marked vertices $M \setminus D$. Note that $R_{v_1} = \{v_3\}$ and $R_{v_2} = \{v_4, v_5\}$.

Lemma 4.4. *The LP relaxation* (2) *for* `UFP-Tree` *has constant integrality gap, and so the above set system is $O(1)$-packable.*

Proof. Consider a feasible solution $y_j, j \in [n]$ to (2). We root the tree T arbitrarily and this naturally defines an ancestor-descendant relationship on the vertices of the tree. The depth of a vertex is its distance from the root. For each path P_j, let v_j be the vertex in P_j with the least depth, and define the *depth of* P_j to be the depth of v_j.

We partition the set of paths into types: $\mathcal{P}_s$, the small paths, are the ones with $s_j \leq \theta/2$, and $\mathcal{P}_l$, the large paths, are the ones with $s_j > \theta/2$. We maintain two feasible sets pf paths, $\mathcal{S}_s$ and $\mathcal{S}_l$, which will be subsets of $\mathcal{P}_s$ and $\mathcal{P}_l$ respectively. We initialize both these sets to be empty. We arrange the paths in ascending order of their depth, and consider them in this order. When we consider a path P_j, we reject it immediately with probability $1 - y_j/4$. If it is not rejected, we consider one of these two cases depending on whether it is small or large: (i) P_j is a small path: we add it to $\mathcal{S}_s$ provided the resulting set $\mathcal{S}_s$ is feasible, i.e., does not violate any vertex capacity; otherwise we discard this path, or (ii) P_j is a large path: we follow the analogous steps with $\mathcal{S}_l$. Finally, we return the better among the two solutions $\mathcal{S}_s$ and $\mathcal{S}_l$.

This completes the description of the rounding algorithm. Now we analyze this algorithm. For the sake of analysis, it will be easier to assume that we pick one of the sets $\mathcal{S}_s$ and $\mathcal{S}_l$ uniformly at random—clearly, this can only hurt the rounding algorithm. We will show that the probability that we pick a path P_j is $\Omega(y_j)$. This will imply the desired result.

We begin with a key observation, whose proof is easy to see.

Observation 4.5. *Suppose a path P_k is considered before another path P_j, and assume $P_j \cap P_k \neq \emptyset$. Then $v_j \in P_k$.*

The following is an easy corollary of the above observation.

Corollary 4.6. *Let P_j be a small path (or large path). Before path P_j is considered, the load on any vertex $v \in P_j$ due to paths in $\mathcal{S}_s$ (or $\mathcal{S}_l$) is at most the load due to these paths on v_j.*

Proof. Assume P_j is a small path (the argument for large paths is identical). Consider a time during the rounding algorithm when P_j has not been considered. For a vertex $v \in P_j$, let F_v be the set of paths in $\mathcal{S}_s$ that contain v. By Observation 4.5, any path in F_v also contains v_j. This implies the claim. □

Corollary 4.6 implies that if we want to check whether adding a path P_j will violate feasibility (of $\mathcal{S}_s$ or $\mathcal{S}_l$), it suffices to check the corresponding load on v_j. So we have the following cases when we consider a path P_j:

- P_j is small: For a path P_k (small or large), we first reject it immediately with probability $1 - y_k/4$—let I_k be the event that it does not get immediately rejected. So, $\Pr[I_k] = y_k/4$. We condition on the event I_j. Let L' be the paths other than y_j which contain v_j and which were not immediately rejected (i.e., paths P_k through v_j for which the event I_k occurs). If the total size of L' is at most $\theta - s_j$, then P_j will get added to $\mathcal{S}_s$. Therefore,

$$\mathbb{P}[P_j \notin \mathcal{S}_s | I_j] \leq \mathbb{P}[s(L') \geq \theta - s_j] = \mathbb{P}\left[\sum_{P_k \in L_{v_j}} s_k I_k \geq \theta - s_j\right]$$
$$\leq \frac{\mathbb{E}[\sum_{P_k \in L_u} s_k I_k^s]}{\theta - s_j} = \frac{\sum_{P_k \in L_u} s_k (y_k/4)}{\theta - s_j} \leq \frac{\theta/4}{\theta - \theta/2} = \frac{1}{2},$$

  where the second last inequality follows from the feasibility of (2) and the fact that P_j is small. Therefore,

$$\mathbb{P}[P_j \in \mathcal{S}_l] = \mathbb{P}[P_j \in \mathcal{S}_s | I_j]\mathbb{P}[I_j] \geq y_k/8.$$

  Since we choose all the paths in $\mathcal{S}_s$ with probability 1/2, probability of P_j getting selected is at least $y_k/16$.

- P_j is large: define the random variables I_k as above. Let L' be the set of large paths other than P_j containing v_j. Clearly, if we select one of the paths in L' in $\mathcal{S}_l$, then P_j cannot be in $\mathcal{S}_l$. But the former event can only happen if we the event I_k occurs for at least one paths P_k in L'. Therefore,

$$\mathbb{P}\left[P_j \notin \mathcal{S}_s | I_j\right] \leq \mathbb{P}\left[\sum_{P_k \in L'} I_k = 1\right]$$
$$\leq \sum_{P_k \in L'} \mathbb{P}[I_k] \leq \sum_{P_k \in L'} \frac{y_k}{4}$$

$$\leq \frac{1}{2}\sum_{P_k\in L'}\frac{s_k y_k}{\theta} \leq \frac{1}{2},$$

where the second last inequality follows from the fact that $s_k \geq \theta/2$ for all paths $P_k \in L'$, and the last inequality follows from the fact that $L' \subseteq L_{v_j}$ and the solution y is feasible to the LP relaxation (2). As in the previous case, we can show that the probability that P_j is selected is at least $y_j/16$.

Thus, we see that for any path P_j, the probability of it getting selected is at least $y_j/16$, and so the desired result follows by linearity of expectation. □

Combining Theorem 2.2 with Lemma 4.4 and 4.3, we get

Corollary 4.7. *There is an $O(\log\log m)$-approximation algorithm for* GenMakespan *when the resources are given by the vertices in a tree and the tasks are given by paths in this tree.*

4.3 Rectangles in the Plane

We now consider the following geometric set system: the tasks are given by a set of n rectangles in the 2-dimensional plane, and the resources are given by the set of all the points in the plane. So, the set L_i for a resource (i.e., point) i is given by the set of rectangles containing i. Since the set of n rectangles will partition the plane into $n^{O(1)}$ connected regions, we can assume without loss of generality that m is polynomially bounded by n. We first show that this set system is λ-safe for a constant value of λ.

Lemma 4.8. *The above mentioned set-system is 4-safe.*

Proof. Let R denote the set of m resources, which are represented by m points in the plane. Let D be a subset of R. Let the points in D be $\{(x_i, y_i)\}_{i=1}^{k}$. Define the set $M := \texttt{Extend}(D)$ to be the Cartesian product of all the x and y coordinates in D, i.e., $M = \{(x_i, y_j) : (x_i, y_i), (x_j, y_j) \in D\}$. Clearly, $|M| \leq k^2$, which satisfies the first condition in the definition of λ-safe. Notice that the points in M correspond to a rectangular grid $\mathcal{G}$ partitioning the plane, where the rectangles on the boundary of $\mathcal{G}$ are unbounded.

Let p be a point in R. We need to define a set $R_p \subseteq M$ such that $L_p \cap L(D) \subseteq L(R_p)$. Let Q denote the minimal rectangle in the grid $\mathcal{G}$ that contains p. Let $q_1, q_2, q_3, q_4 \in M$ denote the corners of rectangle Q (if Q is unbounded then it has fewer than four corners, but the following argument still applies.) Define R_p to be the set of these corner points. Now let J be a task (i.e., rectangle) containing p and a point in D. Then by the construction of M, J must contain at least one of the points in R_p. This proves the lemma. □

We now consider the α-packable assumption. The integrality gap of the packing LP (2) is not well understood for arbitrary rectangles. However, the integrality gap is known to be constant for instances with no "corner" intersections [5], i.e., for any pair T_1, T_2 of rectangles, either $T_1 \supseteq T_2$ or T_1 does not contain any corner of T_2. There is also a constant integrality gap if we allow for a *bi-criteria* approximation. In particular, if the integral solution is allowed to shrink every rectangle by a factor of $(1-\delta)$ for any $\delta > 0$ then there is a polynomial time $poly(\frac{1}{\delta})$-approximation algorithm [1] which rounds the corresponding LP relaxation (2). Theorem 2.2 along with Lemma 4.8 and these results imply the following.

Corollary 4.9. *There is an $O(\log\log n)$-approximation algorithm for* GenMakespan *when the resources are represented by all points in the plane and the tasks are given by a set of n rectangles in the plane, and when one of the following conditions holds:*

- *there are no corner intersections among the rectangles, or*
- *the rectangles chosen in a solution can be shrunk by $(1-\delta)$-factor in either dimension, where $\delta > 0$ is some constant.*

4.4 Fat Objects in the Plane

We generalize the setting of rectangles to more general shapes which are not skewed in any particular dimension. We consider the following set system – the tasks are given by a set of n "fat" objects in a plane and the resources are given by the set of all the points in the plane. For a resource (i.e., point) p, L_p is the set of fat objects containing p. We use the following definition of *fat objects* [8] – a set $\mathcal{F}$ of objects in $\mathbb{R}^2$ is called fat if for every axis-aligned square B of side-length r, we can find a constant number of points $Q(B)$ such that every object in $\mathcal{F}$ that intersects B and has diameter at least r also contains some point in $Q(B)$. Examples of fat objects include squares, disks and triangles/rectangles with constant aspect ratio. For concreteness, one can consider all tasks/objects as disks; note that the radii can be different. Again, although the resources are given by all the points in the plane, it suffices to focus on $m = poly(n)$ many "relevant" points (i.e., resources) given by the set of connected regions formed by these fat objects. We first show that this set-system is λ-safe for a constant value of λ.

Lemma 4.10. *The above-mentioned set system is $O(1)$-safe.*

Proof. Let $\mathcal{F}$ denote the set of fat objects represented by the tasks. Let R be the set of m points in the plane which represent the set of resources, and D be a subset of R. Let $\mathcal{D}$ denote the set of all non-zero pairwise distances between the points in D; note that $|\mathcal{D}| \leq |D|^2$.

We define the set $M := \texttt{Extend}(D)$ as follows: for each point $p \in D$ and distance $\theta \in \mathcal{D}$ let $G(p,\theta)$ be the square centered at p with side-length 10θ. We divide this square into a grid consisting of smaller squares (we call these "cells") of side length 0.1θ. So $G(p,\theta)$ has 100 cells in it. For each cell B in $G(p,\theta)$, add the points $Q(B)$ (from the definition of fat objects with $r := 0.1\theta$) to M.

Clearly, $|M| \leq O(1) \cdot |D|\,|\mathcal{D}| = O(|D|^3) = poly(|D|)$ as required by the first condition of λ-safe. We now check the second condition of this definition. Let p be an arbitrary point in R. We need to show that there is a constant size subset $R_p \subseteq M$ such that $L_p \cap L(D) \subseteq L(R_p) \cap L(D)$.

Let q be the closest point in D to p, and $d(p,q)$ denote the distance between these two points. Note that $d(p,q)$ may not belong to $\mathcal{D}$. We consider the following cases:

- There exists a $\theta \in \mathcal{D}$ with $\frac{d(p,q)}{5} \leq \theta \leq 5d(p,q)$: Consider the grid $G(q,\theta)$. There must be some cell B in this grid that contains p. Define $R_p := Q(B)$, where $Q(B)$ is as in the definition of fat objects (with respect to $\mathcal{F}$).

  Let us see why this definition has the desired properties. Let $F \in \mathcal{F}$ be a fat object which contains p and a point in D. Since q is the closest point in D to p, the diameter of F is at least $d(p,q) > 0.1\theta$, which is also the side length of B. Note that F intersects B because $p \in F$. Therefore, by the definition of $Q(B)$, F must intersect $Q(B)$ as well. Thus, $L_p \cap L(D) \subseteq L(R_p) \cap L(D)$.

- There is no $\theta \in \mathcal{D}$ with $\frac{d(p,q)}{5} \leq \theta \leq 5d(p,q)$: Let $D_0 \subseteq D$ be the subset of D at distance at most $d(p,q)/5$ from q. Let q' be the point in $D \setminus D_0$ which is closest to p. (If $D \setminus D_0 = \emptyset$ then we just ignore all steps involving q' below.) Since $q' \notin D_0$, $d(q,q') > d(p,q)/5$. Moreover, as $\mathcal{D} \cap [\frac{d(p,q)}{5}, 5d(p,q)] = \emptyset$ we have $dist(q,q') > 5d(p,q)$. Using triangle inequality, we get $d(p,q) + d(p,q') \geq d(q,q') > 5d(p,q)$, and so, $d(p,q') > 4d(p,q)$. We are now ready to define R_p. There are two kinds of points in R_p:

- Type-1 points: If D_0 is the singleton set $\{q\}$, we add q to R_p. Otherwise, let $\Delta \in \mathcal{D}$ be maximum pairwise distance between any two points in D_0. Consider the grid $G(q,\Delta)$ – for each cell B in this grid, we add $Q(B)$ to R_p. Note that the number of cells is 100, and so we are adding $O(1)$ points to R_p.
- Type-2 points: Recall that $d(p,q') > 4d(p,q)$. It follows that $d(q,q') \leq d(p,q)+d(p,q') \leq 1.25d(p,q')$, and $d(q,q') \geq d(p,q') - d(p,q) \geq 0.75d(p,q')$. Therefore there is an element $\theta' \in \mathcal{D}$ which lies in the range $[0.75d(p,q'), 1.25d(p,q')]$. We consider the grid $G(q',\theta')$ – there must be a cell in this grid which contains p. Let B be this cell. We add all the points in $Q(B)$ to R_p. Again, we have only added a constant number of points to R_p.

It is clear that R_p is a subset of M. Now we show that it has the desired properties. Let $F \in \mathcal{F}$ be a fat object which contains p and a point in D. Two cases arise:

- $F \cap D_0 \neq \emptyset$: If D_0 is the singleton set $\{q\}$, then F intersects R_p, and we are done. So assume this is not the case. The diameter of F is at least $d(p,q)$. Since the diameter of D_0 is Δ, $G(q,\Delta)$ intersects F, and so there is a cell B in $G(q,\Delta)$ intersecting F. Since the diameter of F is at least $d(p,q) \geq 0.1\Delta$, which is also the side length of B, F must contain a point in $Q(B)$, and so, contains one of the type-1 points in R_p.
- $F \cap D_0 = \emptyset$: Note that F intersects the cell B used in the definition of type-2 points in R_p (because B contains p). Further, the diameter of F is at least $d(p,q')$, which is larger than the side length of B. Therefore, F must contain a type-2 point in R_p.

This completes the proof of the lemma. □

We now check the α-packable condition. Chan and Har-Peled [8] gave an $O(1)$ approximation algorithm for rounding the LP relaxation (2) for such set systems when they have linear "union complexity". Many fat objects, such as disks and rectangles with constant aspect-ratio, are known to satisfy the linear union complexity condition. Some others, e.g. fat triangles have $O(n \log\log n)$ union complexity, which implies $\alpha = O(\log\log n)$. See the survey [3] for more details on union complexity.

Corollary 4.11. *There is an $O(\log\log n)$-approximation algorithm for* GenMakespan *when the resources correspond to all points in the plane and the tasks are given by rectangles with constant aspect-ratio or disks.*

5 Integrality Gap Lower Bounds

We now 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 the underlying deterministic reward maximization problem may have constant integrality gap? For the first question, we show that the integrality gap of our LP relaxation for general set systems is $\Omega\left(\frac{\log m}{(\log\log m)^2}\right)$, and so our LP based approach requires us to put some conditions on the underlying set system. For the second question, we show that even for the special case of set systems given by intervals on a line (as in

Section 4.1), 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 α and λ are constants.

5.1 Lower Bound for Intervals on a Line

We consider the set system as in Section 4.1. Recall that resources are given by m vertices on a line, and tasks by a set of n intervals on the line. We construct such an instance of GenMakespan with $\Omega(\log^* m)$-integrality gap.

Let H be an integer. The line consists of $m = 2^H$ points and $n = 2^{H+1} - 1$ intervals. The intervals are arranged in a binary tree structure. For each "depth" $d = 0, 1, \cdots H$, there are 2^d many disjoint depth-d intervals of width $m/2^d$ each. We can view these intervals as nodes in a complete binary tree $\mathcal{T}$ of depth H where the nodes at depth d correspond to the depth-d intervals, and for any interval I and its parent I' we have $I \subseteq I'$. Moreover, points in the line correspond to root-leaf paths in $\mathcal{T}$ where all intervals in the root-leaf path contain the corresponding point. The size of every depth-d interval j is a random variable $X_j = \mathtt{Ber}(2^{-d})$, i.e. $X_j = 1$ w.p. 2^{-d} and $X_j = 0$ otherwise. The target number of intervals is $t = n$: so we need to select *all* the intervals.

Consider the LP relaxation with a target bound of 1 on the expected makespan. We will show that the LP (6)-(9) is feasible with decision variables $y_j = 1$ for all intervals j. Note that every random variable X_j is already truncated (there is no instantiation larger than one). So constraints (6), (7) and (9) are clearly satisfied.

Lemma 5.1. *For any $K \subseteq [m]$ with $k = |K|$ we have $\sum_{j \in L(K)} \beta_k(X_j) \leq 4k$. Hence, constraint (8) is satisfied with $b = 4$ on the right-hand-side.*

Proof. Consider any subset $K \subseteq [m]$ of vertices on the line. Recall that for any vertex i, L_i denotes the set of intervals that contain it; and $L(K) := \cup_{i \in K} L_i$. We partition $L(K)$ into the following two sets: L' consisting of intervals of depth at most $\log k$ and $L'' = L(K) \setminus L'$ consisting of intervals of depth more than $\log k$. We will bound the summation separately for these two sets.

Bounding the contribution of L'. Note that the total number of intervals of depth at most $\log k$ is less than $2k$. So $|L'| < 2k$. Moreover, $\beta_k(X_j) \leq 1$ for all intervals j. So $\sum_{j \in L'} \beta_k(X_j) \leq |L'| < 2k$.

Bounding the contribution of L''. Consider any vertex $i \in K$. For each depth $d = 0, \cdots H$, L_i contains exactly one interval of depth d. So we have

$$\sum_{j \in L'' \cap L_i} \beta_k(X_j) \leq \sum_{d=\log k}^{H} \beta_k(\mathtt{Ber}(2^{-d})) = \frac{1}{\log k} \sum_{d=\log k}^{H} \log\left(1 + (k-1)2^{-d}\right) \leq \frac{2(k-1)}{\log k} \sum_{d=\log k}^{H} 2^{-d} \leq 2.$$

The first inequality used the facts that (i) L'' contains only intervals of depth more than $\log k$ and (ii) the size of each depth-d interval is $\mathtt{Ber}(2^{-d})$. The second inequality uses $\log(1+x) \leq 2x$ for all $x \geq 0$. It now follows that $\sum_{j \in L''} \beta_k(X_j) \leq \sum_{i \in K} \sum_{j \in L'' \cap L_i} \beta_k(X_j) \leq 2k$.

Combining the two bounds above, we obtain the lemma. $\square$

Next, we show that the expected makespan when all the n intervals are selected is $\Omega(\log^* n)$. To this end, we will show that with constant probability, there is some root-leaf path in $\mathcal{T}$ for which $\Omega(\log^* n)$ random variables in it have size one. Define a sequence $\{h_i\}_{i=0}^{c}$ as follows:

$$h_0 = 2, \quad h_{i+1} - h_i = h_i \cdot 2^{h_i} \text{ for } i = 1, \cdots c-1.$$

We choose $c = \Theta(\log^* H)$ so that $h_c \leq H$.

Lemma 5.2. *For any depth-d interval j, let $\mathcal{I}$ denote the intervals in the subtree of $\mathcal{T}$ below j, from depth d to depth $d + d2^d$. Then* $\Pr\left[\sum_{v\in\mathcal{I}} X_v \geq 1\right] \geq 1 - e^{-d}$.

Proof. We show that $\Pr\left[\sum_{v\in\mathcal{I}} X_v = 0\right] \leq e^{-d}$, which will imply the desired result. Note that for each $h = 0, \ldots, d2^d$, $\mathcal{I}$ contains 2^h intervals at depth $d+h$ and each of these intervals has size given by $\texttt{Ber}(2^{-d-h})$. By independence, the probability that all these sizes are zero is:

$$\prod_{v\in\mathcal{I}} \Pr[X_v = 0] = \prod_{h=0}^{d2^d} (1 - 2^{-d-h})^{2^h} \leq \prod_{h=0}^{d2^d} e^{-2^{-d}} = e^{-d}.$$

□

Using the above result several times yields the desired lower bound.

Lemma 5.3. *With probability at least $\frac{1}{2}$, there is a root-leaf path in $\mathcal{T}$ such that at least c random variables in it are 1.*

Proof. We show the following by induction on i, $0 \leq i \leq c$: with probability at least $\prod_{i'=0}^{i-1}(1-e^{-h_{i'}})$ there is a node v_i at depth h_i such that the root to v_i path has at least i random variables which are 1. For $i = 0$, this follows easily because the root itself is 1 with probability 1. Now assume that the induction hypothesis is true for $i < c$. Let V_i be the set of nodes (i.e., intervals) in $\mathcal{T}$ at depth h_i. For an interval $j \in V_i$, let E_j be the event that j is the first vertex in V_i (say from the left to right ordering) such that the root to j path has at least i random variables which are 1. Let $\mathcal{I}_j$ be the sub-tree of depth $h_i \cdot 2^{h_i}$ below j (so the leaves of $\mathcal{I}_j$ are at depth h_{i+1}); and E'_j be the event that there is a random variable in $\mathcal{I}_j$ which is 1. Lemma 5.2 implies that for any $j \in V_i$, $\Pr[E'_j] \geq (1 - e^{-h_i})$. Since the events E_j are disjoint, and are independent of $E'_{j'}$ for any $j' \in V_i$, we get

$$\Pr[\exists j \in V_i : E_j \wedge E'_j] = \sum_{j\in V_i} \Pr[E_j \wedge E'_j] = \sum_{j\in V_i} \Pr[E_j] \cdot \Pr[E'_j] \geq (1 - e^{-h_i}) \sum_{j\in V_i} \Pr[E_j].$$

Since the events E_j are disjoint, $\sum_j \Pr[E_j] = \Pr[\exists j \in V_i : E_j]$. By induction hypothesis, this probability is at least $\prod_{i'=0}^{i-1}(1 - e^{-h_{i'}})$. Combining the above inequalities, the induction hypothesis follows. Applying the induction hypothesis for $i = c$, we see that the desired event happens with probability at least $\prod_{i=0}^{c-1}(1 - e^{-h_i}) \geq 1 - \sum_{i=0}^{c-1} e^{-h_i} \geq \frac{1}{2}$. □

Combining Lemmas 5.1 and 5.3, we obtain:

Theorem 5.4. *The integrality gap of the LP relaxation given by (6)–(9) for* GenMakespan *when the set system is given by intervals on the line is $\Omega(\log^* m)$.*

5.2 Lower Bound for General Set Systems

Now we consider GenMakespan for general set systems and show that the LP relaxation has $\Omega(\frac{\log m}{(\log\log m)^2})$ integrality gap.

The instance consists of $n = q^2$ tasks and $m = q^q$ resources where q is some parameter. For each task j, the random variable X_j is a Bernoulli random variable that takes value 1 one with probability $\frac{1}{q}$, i.e., the distribution of X_j is $\texttt{Ber}(\frac{1}{q})$. The tasks are partitioned into q groups – $T_1, \cdots T_q$, with q tasks in each group. Each resource is associated with a choice of one task a_j from each group T_j, $j \in [q]$. In other words, the set L_i for any resource i has cardinality q and contains exactly one element from each of the groups T_j. Thus, the total number of resources is q^q. The target number of tasks to be chosen is $n = q^2$, which means every task must be selected.

We first observe that the expected makespan is $\Omega(q)$. Indeed, consider a group T_j. With probability $(1 - 1/q)^q \approx 1/e$, there is a task $a_j \in T_j$ for which the random variable X_{a_j} is 1. So the expected number of tasks for which this event happens is about q/e. As there is a resource associated with *every* choice of one task from each group, it follows that the expected makespan is at least q/e.

Consider the LP relaxation with a target bound of $B = \Theta(\log q)$ on the expected makespan. We will show that the LP constraints (6)-(9) are feasible with decision variables $y_j = 1$ for all objects j. We will scale all the random variables down by a factor of B (because the LP relaxation assumes that the target makespan is 1). Let X denote the scaled Bernoulli r.v. with $X = \frac{1}{B}$ w.p. $\frac{1}{q}$ and $X = 0$ otherwise. Since this random variable will never exceed 1, X' (the truncated part) is same as X, and X'' (the exceptional part) is 0. So constraints (6), (7) and (9) are clearly satisfied. Moreover,

$$\beta_k(X) = \frac{1}{\log k} \log\left(1 + \frac{k^{1/\log q}}{q}\right) \leq \frac{2k^{1/\log q}}{q \log k}, \tag{15}$$

where we used $\log(1 + x) \leq 2x$ for all $x \geq 0$.

Lemma 5.5. *Constraint (8) is satisfied with $b = 8$ for the above instance.*

Proof. Consider any subset $K \subseteq [m]$ of $k = |K|$ resources. Recall that $L(K) \subseteq [n]$ denotes the subset of tasks contained in any of the sets corresponding to K. As every random variable has the same distribution as X, the left-hand-side (LHS) in (8) is just $|L(K)| \cdot \beta_k(X)$. We now consider three cases:

- $k \leq q$. We have $|L(K)| \leq kq$ as each resource is loaded by exactly q tasks. Using (15), the LHS is at most $kq \cdot \beta_k(X) \leq kq \frac{2q^{1/\log q}}{q \log k} \leq 4k$.
- $q < k \leq q^2$. We now use $|L(K)| \leq n = q^2$. By (15), we have $\beta_k(X) \leq \frac{2 \cdot q^{2/\log q}}{q \log k} \leq \frac{8}{q \log k}$. So $LHS \leq q^2 \cdot \beta_k(X) \leq \frac{8q}{\log k} \leq 8k$.
- $k > q^2$. Here we just use $|L(K)| \leq q^2$ and $\beta_k(X) \leq 1$ to get $LHS \leq k$.

The lemma is proved as $LHS \leq 8k$ in all cases. ☐

As $q = \Theta(\frac{\log m}{\log \log m})$, the integrality gap of the LP for GenMakespan is $\Omega\left(\frac{\log m}{(\log \log m)^2}\right)$.

We also show that the integrality gap of LP (2) is at most two for all instances of the deterministic problem that need to be solved in our algorithm. Note that as all sizes are identically distributed, we only need to consider deterministic instances of the reward-maximization problem for which all (deterministic) sizes are identical, say s. We note that LP (2) has an integrality gap $\alpha \leq 2$ on such instances. Using the structure of the above set-system, it is clear that an optimal LP solution will assign the same value $z_i \in [0, 1]$ to all objects in any group G_i. So the LP objective equals $\sum_{i=1}^{q} r(G_i) \cdot z_i$ where $r(G_i)$ is the total reward of the objects in G_i. The congestion constraints imply $\sum_{i=1}^{q} z_i \leq \frac{\theta}{s}$. So this LP now reduces to the max-knapsack problem, which is known to have integrality gap at most two. In particular, choosing all objects in the $\lfloor \frac{\theta}{s} \rfloor$ groups G_i with the highest $r(G_i)$ yields total reward at least half the LP value.

References

[1] A. Adamaszek, P. Chalermsook, and A. Wiese. How to tame rectangles: Solving independent set and coloring of rectangles via shrinking. In *APPROX/RANDOM*, pages 43–60, 2015.

[2] P. K. Agarwal and N. H. Mustafa. Independent set of intersection graphs of convex objects in 2d. *Comput. Geom.*, 34(2):83–95, 2006.

[3] P. K. Agarwal, J. Pach, and M. Sharir. State of the union of geometric objects. In *Surveys in Discrete and Computational Geometry Twenty Years Later*, pages 9–48, 2008.

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

[5] P. Chalermsook. Coloring and maximum independent set of rectangles. In *APPROX-RANDOM*, pages 123–134, 2011.

[6] P. Chalermsook and J. Chuzhoy. Maximum independent set of rectangles. In *SODA*, pages 892–901, 2009.

[7] T. M. Chan. A note on maximum independent sets in rectangle intersection graphs. *Inf. Process. Lett.*, 89(1):19–23, 2004.

[8] T. M. Chan and S. Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012.

[9] C. Chekuri, M. Mydlarz, and F. B. Shepherd. Multicommodity demand flow in a tree and packing integer programs. *ACM Trans. Algorithms*, 3(3):27, 2007.

[10] A. I. Elwalid and D. Mitra. Effective bandwidth of general markovian traffic sources and admission control of high speed networks. *IEEE/ACM Transactions on Networking*, 1(3):329–343, 1993.

[11] A. Gupta, A. Kumar, V. Nagarajan, and X. Shen. Stochastic load balancing on unrelated machines. In *SODA*, pages 1274–1285. Society for Industrial and Applied Mathematics, 2018.

[12] J. Y. Hui. Resource allocation for broadband networks. *IEEE J. Selected Areas in Comm.*, 6(3):1598–1608, 1988.

[13] F. P. Kelly. Notes on effective bandwidths. In *Stochastic Networks: Theory and Applications*, pages 141–168. Oxford University Press, 1996.

[14] J. Kleinberg, Y. Rabani, and E. Tardos. Allocating bandwidth for bursty connections. *SIAM J. Comput.*, 30(1):191–217, 2000.

[15] M. Molinaro. Stochastic ℓ_p load balancing and moment problems via the l-function method. In *SODA*, pages 343–354, 2019.

A Missing Proofs from Section 2

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

(i) for all $X \subseteq [n]$, the set system $(X, \mathcal{L})$ is α-packable and λ-safe, and

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

Proof. For the first statement, consider any $X \subseteq [n]$ and let $\mathcal{L}|_X == \{L_i'\}_{i=1}^m$. The λ-safe is immediate from the definition, by using the same sets M and R_is for each $D \subseteq [m]$; note that $L_i' = L_i \cap X$ for all $i \in [m]$. To see the α-packable property, consider any rewards and sizes $r_j, s_j \geq 0$ for elements $j \in X$, and threshold θ. We extend these rewards and sizes to the entire set $[n]$ by setting $r_j = s_j = 0$ for all $j \in [n] \setminus X$. We now use the fact that the original set-system is α-packable. Let $y \in [0,1]^n$ denote an LP solution to (2). Because $r_j = 0$ for all $j \notin X$, we can set $y_j = 0$ for all $j \notin X$, while not changing the objective. Now, the rounded integer solution $\widehat{y}$ obtains at least a $1/\alpha$ fraction of the LP reward. Moreover, $\widehat{y}$ only selects elements in X as the support of $\widehat{y}$ is contained in the support of y, which is contained in X.

For the second statement, note that the LP constraint matrix in (2) for such a set-system is block-diagonal. Indeed, because of the disjoint union, constraints corresponding to resources in $\mathcal{L}_h$ only involve variables corresponding to X_h, for all $h = 1, \cdots s$. Let $y^{(h)}$ denote the restriction of the LP solution y to elements X_h, for each h. Then, using the α-packable property on $(X_h, \mathcal{L}_h)$, we obtain an integral solution $\widehat{y^{(h)}}$ that has at least a $1/\alpha$ fraction of the reward from $y^{(h)}$. Combining the integer solutions $\widehat{y^{(h)}}$ over all $h = 1, \cdots s$ proves the α-packable property for the disjoint union. $\square$

B Missing Proofs from Section 3

Theorem 3.2 (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.*

Proof. Our algorithm aims to satisfy the constraints (8), but will only achieve the following slightly weaker constraint:

$$\sum_{j \in L(K)} \beta_k(X_j') \cdot y_j \leq \frac{e}{e-1} b \cdot k, \quad \forall K \subseteq [m] \text{ with } |K| = k, \ \forall k = 1, 2, \cdots m, \tag{16}$$

We use the ellipsoid algorithm to find a feasible solution to the above LP. Given $y \in \mathbb{R}^n$ the separation oracle needs to check if constraint (8) is satisfied (the other constraints are easy to check). For each k, $1 \leq k \leq m$, we consider an instance $\mathcal{I}_k$ of the weighted maximum-coverage problem with m sets $\{L_i\}_{i=1}^m$ and weights $w_j = \beta_k(X_j') \cdot y_j$ on each task $j \in [n]$. Note that checking (8) for subsets K of size k is equivalent to checking if the optimal value of $\mathcal{I}_k$ is at most bk. As there is an $\frac{e}{e-1} \approx 1.58$ approximation algorithm for maximum-coverage [], this constraint can be checked approximately. Let $A_k \subseteq [m]$ denote the approximate solution to $\mathcal{I}_k$ that we obtain for each k. Then we have the following cases:

- For some k, the value $\sum_{j \in L(A_k)} \beta_k(X_j') \cdot y_j$ is more than bk. Then, this is a violated constraint, which can be added to the ellipsoid algorithm.

- For each k, the value $\sum_{j \in L(A_k)} \beta_k(X'_j) \cdot y_j$ is at most bk. Then it follows that, for each k, the optimal value of $\mathcal{I}_k$ is at most $\frac{e}{e-1} bk$. This implies that constraint (16) is satisfied.

This proves the desired result. $\square$

Theorem 3.3. *Suppose a set system satisfies the α-packable property. Then there is an $O(\alpha)$-approximation algorithm for* DetCost *relative to the LP relaxation* (10).

Proof. Consider an instance $\mathcal{I}$ of DetCost consisting of a set system $([n], \mathcal{L})$, cost c_j and size s_j for each element $j \in [n]$, and parameters $\theta \geq \max_j s_j$ and $\psi \geq \max_j c_j$. Let y be a solution to the LP (10), with objective function value $T = \sum_j y_j$. We construct an instance $\mathcal{I}'$ of the reward-maximization problem with LP relaxation (2). The set system, sizes of elements and the parameter θ are as in $\mathcal{I}$. Furthermore, the reward r_j of an element j is defined as:

$$r_j := \left(1 - \tfrac{T}{2\psi} c_j\right).$$

Since the set of constraints in (2) is a subset of that in (10), the solution y is also a feasible solution to (2) with objective function value equal to

$$\sum_{j \in [n]} r_j y_j \geq \sum_{j \in [n]} \left(1 - \frac{T}{2\psi} c_j\right) y_j \geq T - T/2 = T/2.$$

The first inequality uses that $c_j \leq \psi$ for all j. Now the α-packable property implies there exists a subset $S \subseteq [n]$ which is a feasible integral solution to (2), whose total reward is at least $\frac{T}{2\alpha}$. Since $r_j \leq 1$ for all j, it follows that $|S| \geq \frac{T}{2\alpha}$ as well.

If the total cost of the elements in S is at most ψ, this is also a feasible solution to $\mathcal{I}$, and the result follows. So assume that $c(S) > \psi$. We greedily partition S into maximal sets $S_1, \ldots, S_u$, each of cost at most ψ; again, we use that each cost is at most ψ. An argument used for the NextFit heuristic for bin packing shows that each consecutive pair of sets must have volume more than ψ, and hence we use at most $\frac{2c(S)}{\psi} + 1 \leq \frac{3c(S)}{\psi}$ bins. Let S^* be the maximum cardinality set among these sets, hence $|S^*| \geq \frac{\psi}{3c(S)}|S|$.

By definition of the rewards r_j, we have that $\sum_{j \in S} r_j = |S| - \frac{T}{2\psi} c(S)$. This must be non-negative (since we would never pick items with negative reward), so we infer $\frac{\psi}{c(S)}|S| \geq \frac{T}{2}$. In turn, we get that $|S^*| \geq \frac{T}{6}$. So in either case, we are guaranteed an $\bar{\alpha} = \max\{2\alpha, 6\} = O(\alpha)$ approximation for DetCost relative to the LP. This proves Theorem 3.3. $\square$