Mathematical Programming (2022) 192:597–630
https://doi.org/10.1007/s10107-021-01741-z

FULL LENGTH PAPER

Series B

Stochastic makespan minimization in structured set systems

Anupam Gupta[1] · Amit Kumar[2] · Viswanath Nagarajan[3] · Xiangkun Shen[4]

Received: 14 May 2020 / Accepted: 8 November 2021 / Published online: 26 November 2021

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, when resources are points and tasks are intervals in a line, we obtain an $O(\log \log m)$-approximation algorithm. 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. Our approach uses a strong LP relaxation using the cumulant generating functions of the random variables. We also show that this LP

A preliminary version appeared in the Proceedings *of the 21st Conference on Integer Programming and Combinatorial Optimization, 2020*. A. Gupta was supported in part by NSF award CCF-1907820, CCF-1955785, and CCF-2006953, and by the Indo-US Joint Center for Algorithms Under Uncertainty. V. Nagarajan and X. Shen were supported in part by NSF grants CCF-1750127, CMMI-1940766, and CCF-2006778.

✉ Viswanath Nagarajan
viswa@umich.edu

Anupam Gupta
anupamg@cs.cmu.edu

Amit Kumar
amitk@cse.iitd.ac.in

Xiangkun Shen
xiangkun.shen@verizonmedia.com

[1] Carnegie Mellon University, Pittsburgh, PA, USA

[2] Indian Instiute of Technology Delhi, New Delhi, India

[3] University of Michigan, Ann Arbor, MI, USA

[4] Yahoo! Research, NYC, New York, NY, USA

has an $\Omega(\log^* m)$ integrality gap, even for the problem of selecting intervals on a line; here $\log^* m$ is the iterated logarithm function.

Keywords Stochastic optimization · Approximation algorithms · Geometric packing · Linear programming

Mathematics Subject Classification 68W25 · 90C27 · 90B15

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 exact algorithm. For general sizes, there is a constant-factor approximation algorithm [5].

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 [11]; 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. [1,5–10].

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 random variables 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 an LP-based α-approximation algorithm for a 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 Sect. 2. To give some intuition for these assumptions, consider intervals on the line. The first assumption holds by the results of [5]. 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 the closest 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 (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 small 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 $poly(\log\log m)$-approximation algorithms 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 [14,17]. 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 very good LP-based approximation algorithms for maximum-weight packing of intervals in a line [5], paths in a tree (with edge loads) [10], rectangles in a plane [7] and fat-objects in a plane [9].

Our techniques draw on prior work on stochastic makespan minimization for identical [17] and unrelated [14] 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 [17] and the use of log-mgf values at multiple scales comes from [14]. The "volume" constraints in our LP also has some similarity to those in [14]: however, a key difference here is that the random variables loading different resources are correlated (whereas they were independent in [14]). Indeed, this is why our LP can only be solved approximately whereas the LP relaxation in [14] 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 [14] obtained a constant-factor approximation. We also prove a super-constant integrality gap in our setting (even for intervals in a line), which rules out the possibility of a constant-factor approximation via our LP.

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 [18]. 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 random variable (r.v.) X_j is known, and these distributions are independent. We assume that all the X_js are discrete r.v.s with polynomial support size. We also assume that each distribution is available explicitly (as a list of realizations and probabilities). In our algorithm, we will use these distributions to compute some "effective" sizes (defined in Sect. 2.2).

For any subset $K \subseteq [m]$ of resources, let $L(K) := \bigcup_{i \in K} L_i$ 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 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 $\alpha \geq 1$:

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

That is, $\sum_j r_j \widehat{y}_j \geq \frac{1}{\alpha} \sum_j r_j y_j$. We also assume, without loss of generality, that the support of $\widehat{y}$ is contained in the support of y. (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 there is a polynomial-time algorithm that, given

any subset $D \subseteq [m]$ of ("dangerous") resources, finds a subset $M \supseteq D$ of ("safe") resources, such that

(a) $|M|$ is polynomially bounded by $|D|$, and
(b) for every $i \in [m]$, there is a subset $R_i \subseteq M$, $|R_i| \leq \lambda$, such that $L_i \cap L(D) \subseteq L(R_i)$; in other words, every task that uses i and some resource from D also uses a resource from R_i.

Recall that $L(D) = \bigcup_{h \in D} L_h$. 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 where set L_i for point $i \in [m]$ consists of the indices of all 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. See Lemmas 14–15 for the proof.
- 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. See Lemmas 16 and 17.

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

Lemma 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 $(X, \mathcal{L}|_X)$ 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 property follows 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 r_j and sizes $s_j \geq 0$ 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 $j \in [n] \backslash X$. 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 $j \notin X$, without 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. □

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 Sect. 4 that the families discussed above satisfy these properties. Our main result is the following:

Theorem 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 $X'_j := X_j \cdot \mathbf{I}_{(X_j \le 1)}$, 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 [17]:

Lemma 2 (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* $\sum_j \mathbb{E}[X''_j] \ge L$ *then* $\mathbb{E}[\max_j X''_j] \ge L/2$.

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) [13–16].

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

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

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 [15]), says that the load is not much higher than the expectation.

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

The usefulness of the effective size comes from a partial converse [17]:

Lemma 4 (Effective Size: Lower Bound) *Let $X_1, X_2, \cdots X_n$ be independent $[0, 1]$ valued r.v.s, and $\{\widetilde{L}_i\}_{i=1}^m$ 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: 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 tasks), 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 deterministic sub-instances.

In order to obtain an $O(\rho)$-approximation algorithm for GenMakespan, it suffices to find a polynomial algorithm that does one of the following:

- find a solution of objective at most ρ, or
- prove that the optimal GenMakespan value is more than 1.

This follows from standard scaling ideas: see Appendix A. Henceforth, we will assume that the optimal value is at most 1, and provide an algorithm that finds a solution with small expected makespan.

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 Sect. 2. We now provide 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 Sect. 2.2.

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*

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

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

for b being a large enough but fixed constant.

Proof The first inequality (4) follows from Lemma 2 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 4 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 4 implies $\sum_{j \in \bigcup_{i \in K} \widetilde{L}_i} \beta_k(X'_j) \le b \cdot k$, where $b = O(1)$ is a fixed constant. Inequality (5) now follows from $\bigcup_{i \in K} \widetilde{L}_i = L(K) \cap S$. □

Lemma 5 allows us to write the following feasibility linear programming relaxation for GenMakespan (assuming the optimal value is at most 1). For every task j, we have a binary variable y_j corresponding to selecting j.

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

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

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

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

In the above LP, $b \ge 1$ denotes the universal constant multiplying k in the right-hand-side of (5). Note that the effective sizes $\beta_k(X_j)$ can be computed in polynomial time because each X_j has polynomial support-size. Despite having an exponential number of constraints, this linear program can be solved approximately in polynomial time. This relies on the ellipsoid algorithm with an approximate separation oracle, see e.g. [4].

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 right-hand-side 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:

$$\sum_{j \in L(K)} \beta_k(X'_j) \cdot y_j \le \frac{e}{e-1} b \cdot k, \quad \forall K \subseteq [m] \text{ with } |K| = k, \ \forall k \in [m], \tag{10}$$

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). To this end, we use the maximum-coverage problem. Given n elements with non-negative weights $\{w_j\}_{j=1}^n$, a collection $\{S_i \subseteq [n]\}_{i=1}^m$ of subsets and bound k, the goal is to select k subsets $T_1, \cdots T_k$ that maximize the total weight $\sum_{j \in \cup_{i=1}^k T_i} w_j$ of covered elements. There is an $\frac{e}{e-1} \approx 1.58$ approximation algorithm for maximum-coverage [12].

For each k, $1 \leq k \leq m$, we consider an instance $\mathcal{I}_k$ of the 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. Let $A_k \subseteq [m]$ denote the approximate solution to $\mathcal{I}_k$ that we obtain for each k by using the algorithm from [12]. 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 (10) is satisfied.

This proves the desired result. □

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 only affects the approximation ratio by a constant factor.

Note that we only use effective sizes β_k of *truncated* r.v.s, so we have $0 \leq \beta_k(X'_j) \leq 1$ for all $k \in [m]$ and $j \in [n]$. Moreover, we make the following assumption (without loss of generality) on the exceptional r.v.s.

Assumption 4 We have $\mathbb{E}[X''_j] \leq 2$ for every task $j \in [n]$.

Indeed, we can simply drop all tasks j with $\mathbb{E}[X''_j] > 2$ as such a task would never be part of an optimal solution- by (4).

3.2 Overview of analysis

Let us give some intuition behind the factor of $O(\log \log m)$ that arises in the approximation ratio. To keep things concrete, consider the special case of intervals on a line: each task is an interval, and each of the m resources is a point on the line. Each task (interval) loads all the resources (points) that lie within the interval. For simplicity, consider the special case where we have an *integral* solution y to the LP relaxation (6)–(9), and therefore there is no need to perform any rounding. (Our analysis loses a $\log \log m$ factor even in this special integral case.) Let T denote the set of intervals for which $y_j = 1$. We would like to argue that the expected makespan due to selecting set T is $O(\log \log m)$.

To this end, we partition the points into $O(\log \log m)$ groups such that (roughly speaking) the expected makespan due to each group is $O(1)$. We maintain a variable k which is initialized to 2, and a set J of remaining intervals (initially equal to T). Consider a greedy procedure to build an ordering $i_1, i_2, \ldots, i_m$ on the points as follows.

Given $i_1, \ldots, i_r$, define i_{r+1} to be any point i for which $\sum_{j \in J \cap L_i} \beta_k(X'_j) > b$, and remove all intervals containing this point i from J. If there is no such point then we update k to k^2, and continue. (If k exceeds m, order the remaining points arbitrarily.) Observe that k takes values which are of the form $k_\ell := 2^{2^\ell}$ for non-negative integers ℓ. We refer to the index ℓ as the "class". For each class ℓ, let D_ℓ denote the set of points in the above sequence that were added when k was equal to k_ℓ. Note that $|D_\ell| \leq k_\ell$—this follows directly from (8). Indeed, if $|D_\ell| > k_\ell$, then by choosing any k_ℓ points from D_ℓ, constraint (8) (with $k = k_\ell$) would not be satisfied. For each class ℓ, let $J_\ell \subseteq T$ denote those intervals that were removed from J when $k = k_\ell$.

We now argue about the makespan of D_ℓ (i.e., the class-ℓ points) due to J_ℓ (the class-ℓ intervals). We first observe that for any point $i \in D_\ell$, $\sum_{j \in J_\ell \cap L_i} \beta_{k_{\ell-1}}(X'_j)$ is at most b. Indeed, if not, this point i would have been added to $D_{\ell-1}$ instead. We now apply Lemma 3: for any point $i \in D_\ell$, the probability that intervals J_ℓ load i to more than $b + 4$ is at most $k_{\ell-1}^{-4} = k_\ell^{-2}$. Then, by a union bound over all points in D_ℓ, the probability that intervals in J_ℓ load *any* point in D_ℓ to more than $b + 4$ is at most $|D_\ell| \cdot k_\ell^{-2} \leq \frac{1}{k_\ell}$. With some additional work, we can also show that the expected makespan of D_ℓ (due to intervals J_ℓ) is a constant. These arguments are formalized (for the general setting) in Lemmas 6 and 7.

However, for any particular point $i \in D_\ell$, we also need to worry about intervals which are in $(T \setminus J_\ell) \cap L_i$. These intervals must belong to previous classes, by the construction of the ordering. For each class $\ell' < \ell$, consider the two points in $D_{\ell'}$ closest to i on either side: this gives us at most $2 \log \log m$ such "representative" points. Any interval in $(T \setminus J_\ell) \cap L_i$ would load at least one of these representatives. Hence, we can bound the load from these intervals by the total load on the representatives, which is $O(\log \log m)$ in expectation. This is formalized by the λ-safe property and Lemma 11.

In this overview, we omitted the issue of rounding the LP solution. This is handled by classifying tasks as being large/small based on their y_j value (see Lemma 9) and using the α-packable property (see Lemmas 8 and 10).

3.3 The deterministic subproblem

We actually need a slight 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. We use the following LP relaxation:

$$\max \sum_{j=1}^{n} y_j$$

$$s.t. \sum_{j \in S} s_j \cdot y_j \leq \theta, \qquad \forall S \in \mathcal{S}$$

$$\sum_{j\in[n]} c_j \cdot y_j \le \psi,$$
$$0 \le y_j \le 1, \qquad \forall j \in [n] \tag{11}$$

The following result, which motivates the α-packable property, shows that the α-packable property for a set system implies an $O(\alpha)$-approximation for the DETCOST problem.

Theorem 5 (detsolve) *Suppose a set system satisfies the α-packable property. Then there is an $O(\alpha)$-approximation algorithm for* DETCOST *relative to the LP relaxation* (11).

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 $\theta \ge \max_j s_j$ and $\psi \ge \max_j c_j$. Let y be a solution to the LP (11), 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 (11), the solution y is also a feasible solution to (2) with objective function value equal to

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

The inequality uses the fact that $\sum_{j=1}^{n} c_j y_j \le \psi$. Now the α-packable property implies that we can find a subset $S \subseteq [n]$ which is a feasible integral solution to (2), whose total reward $\sum_{j\in S} r_j \ge \frac{T}{2\alpha}$. Since $r_j \le 1$ for all j, it follows that $|S| \ge \frac{T}{2\alpha}$ as well. Moreover, by definition of r_j, we have that $\sum_{j\in S} r_j = |S| - \frac{T}{2\psi} c(S) \ge \frac{T}{2\alpha} \ge 0$. Hence,

$$|S| \ge \frac{c(S)}{2\psi} T. \tag{12}$$

If the total cost of the elements in S is at most ψ, this is also a feasible solution to $\mathcal{I}$ with $|S| \ge \frac{T}{2\alpha}$.

Below, we assume that $c(S) = \sum_{j\in S} c_j > \psi$. Starting with a partition of S into singletons, we repeatedly merge any two parts whose total cost is at most ψ. Let $S_0, \ldots, S_{u-1}$ denote the parts at the end of this process. As each element has cost at most ψ and we only merge parts when their total cost is at most ψ, it follows that the cost $c(S_k)$ of each part S_k is at most ψ. Moreover, the total cost of *any* pair of parts is more than ψ. This implies that $c(S_k) + c(S_{k+1}) > \psi$ for each $k = 0, \ldots u-1$ (the indices are modulo u). Adding these u inequalities, we have $2\sum_{k=0}^{u-1} c(S_k) > u \cdot \psi$ which implies $u < \frac{2}{\psi} \sum_{k=0}^{u-1} c(S_k) = \frac{2}{\psi} c(S)$. Let S^* be the maximum cardinality set among $\{S_k\}_{k=0}^{u-1}$. Note that $|S^*| \ge \frac{1}{u}|S| > \frac{\psi}{2c(S)}|S|$. Using (12) we obtain $|S^*| \ge \frac{T}{4}$.

So in either case, we are guaranteed an $\bar{\alpha} = \max\{2\alpha, 4\} = O(\alpha)$ approximation for DETCOST relative to the LP. This completes the proof. □

3.4 Rounding a feasible LP solution

We first give some intuition about the GENMAKESPAN 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 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 $O(\alpha\lambda)$ from each deterministic instance, which leads to an overall $O(\alpha\lambda \log\log m)$-approximation ratio. The procedure is described formally in Algorithm 1.

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-ℓ 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 $D_\ell \leftarrow D_\ell \bigcup \{i\}$;
7 Set $\widetilde{L}_i \leftarrow J \cap L_i$ and $J \leftarrow J \backslash \widetilde{L}_i$;
8 Define the class-ℓ tasks $J_\ell \leftarrow \bigcup_{i \in D_\ell} \widetilde{L}_i$;
9 Use λ-safety on the set system $(J_\ell, \{L_i \cap J_\ell\}_{i \in [m]})$ to get $M_\ell := \texttt{Extend}(D_\ell)$;
10 $\rho \leftarrow 1 + \log\log m$;
11 Define class-ρ tasks $J_\rho = J$ and class-ρ resources $M_\rho := D_\rho = [m] \backslash \left(\bigcup_{\ell=0}^{\rho-1} D_\ell \right)$;
12 Define an instance $\mathcal{C}$ of DETCOST as follows: the set system is the disjoint union of the set systems (J_ℓ, M_ℓ) for $\ell = 0, \ldots, \rho$. The other parameters are as follows:

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

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

where $\bar{\alpha}$ is the approximation ratio from Theorem 5 ;
13 Let $N_H = \{j \in [n] : y_j > 1/\bar{\alpha}\}$;
14 Let $\bar{y}_j = \bar{\alpha} \cdot y_j$ for $j \in [n] \backslash N_H$ and $\bar{y}_j = 0$ otherwise ;
15 Round $\bar{y}$ (as a feasible solution to (11)) using Theorem 5 to obtain N_L ;
16 Output $N_H \bigcup N_L$.

The algorithm proceeds in $\log\log m$ iterations of the **for** loop in Lines 3–9. The set J denotes the remaining tasks at any point in the algorithm. In each iteration ℓ, we make use of effective sizes β_k with parameter $k = 2^{2^\ell}$ (see Line 3). 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 and we only consider the remaining tasks J. The set of such resources is grouped in the set D_ℓ (called the class-ℓ resources). We also define the class-ℓ tasks J_ℓ to be all remaining tasks (in J) which can load the resources D_ℓ. 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_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: in Line 9 we expand D_ℓ to a larger set of resources $M_\ell := \texttt{Extend}(D_\ell)$, which will be used to bound the load induced by J_ℓ on resources outside D_ℓ. We use (J_ℓ, M_ℓ) to represent the set system corresponding to class-ℓ: note that each set is of the form $L_i \cap J_\ell$ for some $i \in M_\ell$.

Having partitioned the tasks into classes $J_1, \ldots, J_\rho$, we consider the disjoint union $\mathcal{D}$ of the set systems (J_ℓ, M_ℓ), for $\ell = 1, \ldots, \rho$. While the sets D_ℓ are disjoint, the sets M_ℓ may not be disjoint. For each resource appearing in multiple sets M_ℓ, we make distinct copies in the combined set-system $\mathcal{D}$. Then we set up an instance $\mathcal{C}$ of DetCost (in Line 12): the set system is $\mathcal{D}$, 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 12. In Line 13, we include into our solution, all tasks (N_H) that have a large LP value. Then, we define a scaled-up fractional solution $\bar{y}$ (in Line 14) supported on all other tasks $[n]\backslash N_H$: we will show later that this is feasible to the LP relaxation (11) for $\mathcal{C}$. Finally, we use Theorem 5 to round $\bar{y}$ to an integral solution N_L (in Line 15) which is added to our solution.

3.5 The analysis

We now show that the expected makespan for the solution produced by the rounding algorithm above is $O(\alpha\lambda\rho)$, where $\rho = 1 + \log\log m$ is the number of classes. In particular, we show that the expected makespan (taken over all resources) due to the selected tasks from each class ℓ is $O(\alpha\lambda)$.

Our first lemma shows that the fractional load on every resource due to class-ℓ tasks (using effective size $\beta_{2^{2^\ell}}$) is at most a constant.

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

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

Proof If $\ell = 0$, we have $r = 2$. Using the LP constraint (8) for a subset $\{i, i'\}$ of size two containing the resource i, we have:

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

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., $J = \bigcup_{\ell' \geq \ell} J_{\ell'}$. The terminating condition in Line 5 (for iteration $\ell - 1$) implies that

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

which implies the lemma. □

Next, we bound the sizes of sets D_ℓ and M_ℓ as functions of ℓ.

Lemma 7 *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 lemma is trivial for the last class $\ell = \rho$ as $k \geq m$ in this case. Now consider any class $\ell < \rho$. Using the condition in Line 5, we have:

$$\sum_{j \in \widetilde{L}_i} \beta_{k^2}(X'_j) \cdot y_j > 2b, \quad \forall i \in D_\ell, \tag{13}$$

where $\widetilde{L}_i$ is as defined in Line 7. Note that 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 $|D_\ell| > k^2$. Letting $K \subseteq D_\ell$ be any set of size k^2, we have:

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

which is a contradiction. Above, the first inequality uses (13) and $K \subseteq D_\ell$, and the last inequality uses the LP constraint (8) on subset K. This proves the first part of the lemma. Finally, the λ-safe property implies that $|M_\ell|$ is polynomially bounded by $|D_\ell|$, which proves the second part. □

We now show that the fractional solution $\bar{y}$ from Line 14 is feasible to the LP relaxation for DETCOST given in (11).

Lemma 8 *The fractional solution $\bar{y}$ is feasible for the LP relaxation (11) corresponding to the* DETCOST *instance $\mathcal{C}$. Moreover, we have $\max_j s_j \leq \theta$ and $\max_j c_j \leq \psi$ in instance $\mathcal{C}$.*

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:

$$\sum_{j \in J_\ell \cap L_i} \beta_k(X'_j) \cdot \bar{y}_j \le \theta = 2\bar{\alpha}b, \tag{14}$$

where $k = 2^{2^\ell}$. Since $\bar{y} \le \bar{\alpha} \cdot y$, this follows directly from Lemma 6.

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

Based on this lemma, we can indeed apply Theorem 5 to round $\bar{y}$ into an integer solution (as done in Line 15). We now analyze our solution $N_H \bigcup N_L$. Recall that N_H consists of all tasks j with $y_j > 1/\bar{\alpha}$ and N_L is the rounded solution obtained from $\bar{y}$.

Lemma 9 *The solution obtained in Algorithm 1 has* $|N_H| + |N_L| \ge t$.

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

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

which completes the proof. □

We now bound the expected makespan of our solution $N := N_H \bigcup N_L$. We will focus on a particular class $\ell \le \rho$ 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 $N_\ell := N \cap J_\ell$ be the selected class-ℓ tasks, 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 the selected class-ℓ tasks. The following lemma can be viewed as the "rounded" version of Lemma 6.

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

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

Proof Since $N_\ell \cap L_i = (N_H \cap J_\ell \cap L_i) \bigcup (N_L \cap J_\ell \cap L_i)$, we bound the left-hand-side above in two parts. By Lemma 6, the solution y has $\sum_{j \in J_\ell \cap L_i} \beta_k(X'_j) \cdot y_j \le 2b$. As each task $j \in N_H$ has $y_j > 1/\bar{\alpha}$,

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

Since N_L is a feasible integral solution to (11), the size constraint for $i \in M_\ell$ implies that

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

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 11 *For any class $\ell \le \rho$, we have $\mathbb{E}\left[\max_{i \in M_\ell} \mathit{Load}_i^{(\ell)}\right] \le 4\bar{\alpha} b + O(1)$ and therefore, $\mathbb{E}\left[\max_{i=1}^m \mathit{Load}_i^{(\ell)}\right] \le 4\lambda\bar{\alpha} b + O(\lambda) = O(\alpha\lambda)$.*

Proof Consider a resource $i \in M_\ell$. Lemmas 3 and 10 imply that for any $\gamma > 0$,

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

By a union bound, we get

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

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

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

which completes the proof of the first statement.

We now prove the second statement. Consider any class $\ell < \rho$: by definition of J_ℓ, we know that $J_\ell \subseteq L(D_\ell)$. The λ-safe property implies that for every resource $i \in [m]$ there is a subset $R_i \subseteq M_\ell$ with $|R_i| \le \lambda$ and $L_i \cap L(D_\ell) \subseteq L(R_i)$; using $J_\ell \subseteq L(D_\ell)$ the latter property implies $L_i \cap J_\ell \subseteq L(R_i) \cap J_\ell$. Because $N_\ell \subseteq J_\ell$, we also have $L_i \cap N_\ell \subseteq L(R_i) \cap N_\ell$. Therefore,

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

Taking expectation on both sides and using the first statement in the lemma, we obtain the desired result.

Finally, for the last class $\ell = \rho$, note that any task in J_ρ loads only the resources in $D_\rho = M_\rho$. Therefore, $\max_{i=1}^{m} \mathsf{Load}_i^{(\ell)} = \max_{z \in M_\ell} \mathsf{Load}_z^{(\ell)}$. Taking expectation on both sides, we obtain the second statement. □

Using Lemma 11, we can bound the expected makespan due to all truncated random variables:

$$\mathbb{E}\left[\max_{i=1}^{m} \sum_{j \in N \cap L_i} X'_j\right] = \mathbb{E}\left[\max_{i=1}^{m} \sum_{\ell=0}^{\rho} \mathsf{Load}_i^{(\ell)}\right] \le \sum_{\ell=0}^{\rho} \mathbb{E}\left[\max_{i=1}^{m} \mathsf{Load}_i^{(\ell)}\right] \le O(\alpha\lambda\rho). \tag{15}$$

The next lemma handles exceptional random variables.

Lemma 12 $\mathbb{E}\left[\sum_{j \in N} X''_j\right] = \sum_{j \in N} c_j \le 4\bar{\alpha}$.

Proof Feasibility of constraint (7) implies that $\sum_{j=1}^{n} c_j \cdot y_j \le 2$. As each task $j \in N_H$ has $y_j > 1/\bar{\alpha}$, we have $\sum_{j \in N_H} c_j \le 2\bar{\alpha}$. For tasks in N_L, the fact that N_L is a feasible integral solution to (11) implies that $\sum_{j \in N_L} c_j \le \psi = 2\bar{\alpha}$. This completes the proof. □

Finally, using (15) and Claim 12, we have:

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

This completes the proof of Theorem 2.

4 Applications

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

4.1 Intervals on a line

We are given a path graph on n vertices, which we call a line. The resources are the vertices in this line. Each task corresponds to an interval in this line and loads all the vertices in the corresponding interval. For each vertex i, L_i denotes the subset 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 [5]—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 13 *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 $L_i \cap L(D) \subseteq L(R_i)$. This is easy to see. Consider a task j (represented by 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, j belongs to $L(R_i)$ as well. □

Theorem 2 now implies the following.

Corollary 1 *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 $|V| = m$ vertices, and a set of n paths, $\{P_j\}_{j=1}^{n}$, in this tree. The resources correspond to vertices and the tasks correspond to paths. For a vertex $i \in [m]$, L_i is the set of paths which contain i. We first show the λ-safe property.

Lemma 14 *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 $L_i \cap L(D) \subseteq L(R_i)$. 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 Fig. 1. We claim that:

$$L_i \cap L(D) \subseteq L_{v_i} \cap L(D) \tag{16}$$

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

- If $v_i \in M$, we set $R_i = \{v_i\}$. By (16) we have $L_i \cap L(D) \subseteq L_{v_i} = L(R_i)$.
- 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 $R_i := \{a_i, b_i\}$. Let Q be the path from a_i to b_i in T'. Observe that Q contains v_i, all internal vertices in Q have degree 2 (in T') and $Q \cap M = \{a_i, b_i\}$. Let P_j be any path which contains i and a vertex w in D. By (16) $v_i \in P_j$. The part of P_j from v_i to w must lie in T' and hence contains either a_i or b_i.

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

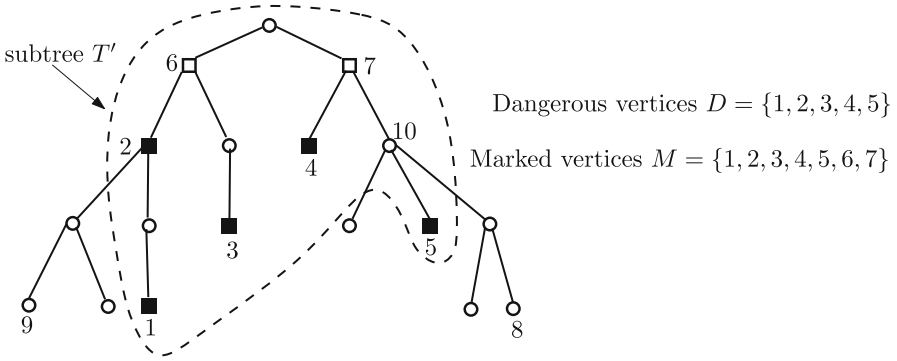


Fig. 1 The solid-square vertices are the "dangerous" vertices D. The box vertices are the additional marked vertices $M \setminus D$. For vertex 8, we have $v_8 = 10$ and $R_8 = \{5, 7\}$. Similarly, for vertex 9, $v_9 = 2$ and $R_9 = \{2\}$

We now consider the α-packable property. As in the case of the line graph application, this is equivalent to 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. [10], and our rounding algorithm is inspired by their approach.

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.

Lemma 15 *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 $\{y_j\}_{j=1}^n$ 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 of paths, $\mathcal{S}_s \subseteq \mathcal{P}_s$ and $\mathcal{S}_l \subseteq \mathcal{P}_l$. We initialize both $\mathcal{S}_s, \mathcal{S}_l = \emptyset$. We consider the paths in ascending order of depth. Each path P_j is rejected immediately with probability $1 - y_j/4$ and with the remaining $\frac{y_j}{4}$ probability we do the following: if P_j is a small (resp. large) path, we add it to $\mathcal{S}_s$ (resp. $\mathcal{S}_l$) provided the resulting set $\mathcal{S}_s$ (resp. $\mathcal{S}_l$) is feasible, i.e., it does not violate any vertex capacity. Finally, we return the better among the two solutions $\mathcal{S}_s$ and $\mathcal{S}_l$.

For the analysis, we will show that

$$\mathbb{P}\left[P_j \in \mathcal{S}_s \bigcup \mathcal{S}_l\right] \geq \frac{y_j}{8}, \quad \forall j \in [n]. \tag{17}$$

This would imply the lemma because our solution's expected objective is:

$$\mathbb{E}\left[\max\left\{\sum_{j:P_j\in\mathcal{S}_s} r_j, \sum_{j:P_j\in\mathcal{S}_l} r_j\right\}\right] \geq \frac{1}{2}\sum_{j=1}^{n} r_j \cdot \mathbb{P}\left[P_j \in \mathcal{S}_s \bigcup \mathcal{S}_l\right] \geq \frac{1}{16}\sum_{j=1}^{n} r_j \cdot y_j.$$

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

Observation 6 *Suppose that path P_k is considered before another path P_j and $P_j \cap P_k \neq \emptyset$. Then $v_j \in P_k$.*

Observation 7 *Let P_j be a small(resp. large) path. Before path P_j is considered, the load on any vertex $v \in P_j$ due to paths in $\mathcal{S}_s$ (resp. $\mathcal{S}_l$) 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 before P_j is 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 6, any path in F_v also contains v_j. This implies the claim. □

Observation 7 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 (as all capacities are uniform). We are now ready to prove (17). For any path P_k (small or large), let I_k be the indicator of the event that P_k does not get immediately rejected; so $\mathbb{P}[I_k] = y_k/4$. We consider two cases:

– P_j is small. We condition on the event $I_j = 1$: note that $\mathbb{P}[P_j \in \mathcal{S}_s] = \mathbb{P}[I_j = 1] \cdot \mathbb{P}[P_j \in \mathcal{S}_s | I_j = 1]$. Let $L' \subseteq [n]$ denote the indices of paths P_k considered before P_j with $v_j \in P_k$ and $I_k = 1$. Note that $L' \subseteq L_{v_j}$. If the total size of L' is at most $\theta - s_j$, then P_j will get added to $\mathcal{S}_s$ (conditioned on $I_j = 1$). So,

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

where the last inequality follows from LP constraints in (2) and the fact that P_j is small. Therefore,

$$\mathbb{P}[P_j \in \mathcal{S}_s] = \mathbb{P}[P_j \in \mathcal{S}_s | I_j = 1] \cdot \mathbb{P}[I_j = 1] \geq y_k/8.$$

– P_j is large. Let L'' denote the indices of the large paths P_k considered before P_j with $v_j \in P_k$. If none of the paths indexed L'' is selected then P_j will be added to $\mathcal{S}_l$. Moreover, path P_k can be selected only if $I_k = 1$. So,

$$\mathbb{P}\left[P_j \notin \mathcal{S}_l | I_j = 1\right] \leq \mathbb{P}\left[\sum_{k \in L''} I_k \geq 1\right] \leq \sum_{k \in L''} \mathbb{P}[I_k = 1] \leq \sum_{k \in L''} \frac{y_k}{4}$$
$$\leq \frac{1}{2} \sum_{k \in L''} \frac{s_k y_k}{\theta} \leq \frac{1}{2},$$

where the second last inequality follows from the fact that $s_k \geq \theta/2$ for all $k \in L''$, and the last inequality follows from the fact that $L'' \subseteq L_{v_j}$ and the LP constraints (2). As in the previous case, this implies $\mathbb{P}[P_j \in \mathcal{S}_l] \geq \frac{y_j}{8}$.

This completes the proof of (17) and the lemma. □

Combining Theorem 2 with Lemmas 15 and 14, we get

Corollary 2 *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 Axis-aligned rectangles in the plane

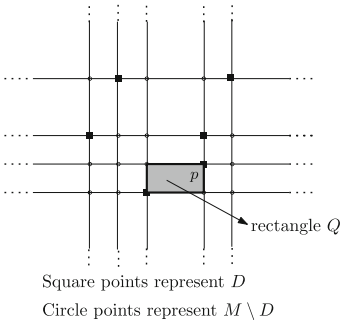
We now consider the following geometric set system: the tasks are n axis-aligned rectangles in the plane and the resources are all points in the plane. The set L_i for a resource (i.e., point) i is given by the set of rectangles containing i. Note that any set of n rectangles partitions the plane into $poly(n)$ many connected regions: this follows from the fact that the total number of intersection points is $O(n^2)$. We designate one point in each connected region as the representative point for that region. Clearly, it suffices to bound the loads on the representative points. Note that the number of representative points is $m = poly(n)$. Below, whenever we refer to an arbitrary point p, it is equivalent to using p's representative point.

Lemma 16 *The above mentioned set-system is 4-safe.*

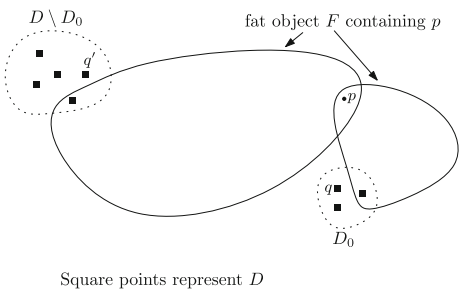
Proof Let $D = \{(x_i, y_i)\}_{i=1}^{k}$ be a subset of points. Define the set $M := \texttt{Extend}(D)$ to be the Cartesian product of all the x and y coordinates in D, i.e., $M = \{(x_i, y_j) : (x_i, y_i), (x_j, y_j) \in D\}$. Clearly, $|M| \leq k^2$, which satisfies the first condition in the definition 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. See Fig. 2a.

Let p be any point. 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 $R_p \subseteq M$ 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. By construction of M, it must be that J contains one of the points in R_p. This proves the lemma. □

We now consider the α-packable assumption. Corollary 5 in Appendix B proves that this set-system is $O((\log \log n)^2)$-packable. Therefore, using Theorem 2 we obtain:



(a) The rectangular grid $\mathcal{G}$ in Lemma 16.



(b) Example for Case 2 in Lemma 17.

Fig. 2 Examples for rectangles and fat objects

Corollary 3 *There is an* $O\left((\log\log n)^3\right)$*-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 axis-aligned rectangles.*

4.4 Fat objects in the plane

We now consider more general shapes which are not skewed in any particular direction. The tasks are given by a set of n "fat" objects in a plane and the resources are given by the set of all points in the plane. We assume that the number of intersection points between any pair of objects is constant, which is true for all our specific applications (disks, triangles, rectangles). This implies that any set of n objects partitions the plane into $m = poly(n)$ many connected regions. As in Sect. 4.3, we designate one point in each connected region as the representative point for that region and focus on the loads of the m representative points. Whenever we refer to an arbitrary point p, it is equivalent to using p's representative point. For any resource (i.e., point) p, L_p is the set of fat objects containing p.

Definition 2 *(Fat objects* [9]*)* 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 (with arbitrary diameters) and triangles/rectangles with constant aspect ratio (i.e., when the ratio of the maximum to minimum side length is constant). For concreteness, one can consider all objects to be disks; note that the radii can be different.

Lemma 17 *The above-mentioned set system is* $O(1)$*-safe.*

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

We define the set $M := \texttt{Extend}(D)$ as follows: for each point $p \in D$ and distance $\theta \in \mathcal{H}$ 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 (called *cells*) of side length 0.1θ. So $G(p, \theta)$ has 100 cells in it. For each cell B in $G(p, \theta)$, add to M the points $Q(B)$ from Definition 2 with $r := 0.1\theta$.

Clearly, $|M| \leq O(1) \cdot |D|\,|\mathcal{H}| = O(|D|^3) = poly(|D|)$ as required by the first condition of λ-safe. We now check the second condition of this definition. Let p be an arbitrary point. We need to show that there is a constant size subset $R_p \subseteq M$ such that $L_p \cap L(D) \subseteq L(R_p)$. Let q be the closest point in D to p, and $d(p, q)$ denote the (Euclidean) distance between these two points. Note that $d(p, q)$ may not belong to $\mathcal{H}$. We consider the following cases:

Case 1: there exists some $\theta \in \mathcal{H}$ 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 Definition 2 (with respect to $\mathcal{F}$).

Let us see why this definition has the desired property. Let $F \in \mathcal{F}$ be any object which contains p and some point $r \in D$. Since q is the closest point in D to p, the diameter of F is at least $d(p, r) \geq d(p, q) > 0.1\theta$, which is the side length of B. Note also that F intersects B because $p \in F$. So, by Definition 2, the object F must intersect $Q(B)$ as well. Thus, $L_p \cap L(D) \subseteq L(R_p)$.

Case 2: there is no $\theta \in \mathcal{H}$ 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; see Fig. 2b. (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 $\mathcal{H} \cap [\frac{d(p,q)}{5}, 5d(p, q)] = \emptyset$ we have $d(q, q') > 5d(p, q)$. 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 $\{q\}$, add q to R_p. Otherwise, let $\Delta \in \mathcal{H}$ be maximum pairwise distance between any two points in D_0. Note that:

$$\Delta = \max_{q_1, q_2 \in D_0} d(q_1, q_2) \leq \max_{q_1, q_2 \in D_0} (d(q, q_1) + d(q, q_2)) \leq \frac{2}{5} d(p, q).$$

  For each cell B in the grid $G(q, \Delta)$, add $Q(B)$ to R_p. Note that the number of cells is 100, and so we only add $O(1)$ many 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 $d(q, q') \geq d(p, q') - d(p, q) \geq 0.75d(p, q')$. So there is an element $\theta' \in \mathcal{H}$ with $0.75d(p, q') \leq \theta' \leq 1.25d(p, q')$. We consider the grid $G(q', \theta')$—there must be a cell in this grid which contains p. Let B be this cell. Add all the points in $Q(B)$ to R_p. Again, we only add a constant number of points to R_p.

It is clear that R_p is a subset of M. Now, consider any object $F \in \mathcal{F}$ which contains p and some point in D. We will show that F also contains some point in R_p, which would prove $L_p \cap L(D) \subseteq L(R_p)$. Two cases arise:

- $F \cap D_0 \neq \emptyset$: If $D_0 = \{q\}$, then F clearly intersects R_p. So assume that $|D_0| \geq 2$. Recall that Δ is the diameter of D_0. So, the grid $G(q, \Delta)$ contains all of D_0, which implies that there is a cell B in $G(q, \Delta)$ intersecting F. As q is the closest point in D to p, the diameter of F is at least $d(p, q) \geq 0.1\Delta$, the side length of B. Hence, by Definition 2, F must contain a point in $Q(B)$, and so, contains one of the type-1 points in R_p.
- $F \cap D_0 = \emptyset$: Recall point q' and value θ' used in the definition of type-2 points in R_p. Note that there is some cell B in $G(q', \theta')$ that contains p; so object F intersects cell B. Further, F contains some point $r \in D \setminus D_0$ which implies that the diameter of F is at least $d(p, r) \geq d(p, q') \geq 0.8 \cdot \theta'$, which is larger than the side length of B. So, by Definition 2, F must contain a point in $Q(B)$, i.e., some type-2 point in R_p.

This completes the proof of the lemma. □

For the α-packable condition, Corollary 6 in Appendix B implies that disks (of arbitrary radii) are $O(\log \log n)$-packable. And, Corollary 7 implies that fat triangles are $O(\log^* n \cdot \log \log n)$-packable. Combined with Theorem 2 and Lemma 17, we obtain:

Corollary 4 *The* GENMAKESPAN *problem admits an* $O\left((\log \log n)^2\right)$*-approximation algorithm when tasks are disks in the plane, and an* $O\left((\log^* n) \cdot (\log \log n)^2\right)$*-approximation algorithm when tasks are fat triangles in the plane.*

5 Integrality gap lower bounds

We now study the limitations of our LP relaxation (6)–(9). There are two natural questions—(i) can we obtain an $O(1)$-approximation for GENMAKESPAN under the α-packable and λ-safe assumptions with $\alpha, \lambda = O(1)$? and (ii) can we obtain an approximation ratio similar to Theorem 2 without the λ-safe assumption? For the first question, we show that even for set systems given by intervals on a line (as in Sect. 4.1) where $\alpha, \lambda = O(1)$, the integrality gap of our LP is $\Omega(\log^* m)$. For the second question, we show that the integrality gap of our LP for general set systems is $\Omega\left(\frac{\log m}{(\log \log m)^2}\right)$. So we cannot get an approximation ratio that is significantly better than logarithmic without some additional condition (such as λ-safe) on the set system.

5.1 Lower bound for intervals on a line

We consider the set system as in Sect. 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 $X_j = \texttt{Ber}(2^{-d})$, 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 18 *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 $L(K) := \bigcup_{i \in K} L_i$. We partition $L(K)$ into the following two sets: L' consisting of intervals of depth at most $\log k$ and $L'' = L(K) \backslash L'$ 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

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

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 $\texttt{Ber}(2^{-d})$. 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. □

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 19 *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,*

$$\mathbb{P}\left[\sum_{v\in\mathcal{I}} X_v \geq 1\right] \geq 1 - e^{-d}.$$

Proof We show that $\mathbb{P}\left[\sum_{v\in\mathcal{I}} X_v = 0\right] \leq e^{-d}$, 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:

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

which proves the lemma. □

Lemma 20 *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, $0 \leq i \leq c$:

$$\text{with probability at least } \prod_{i'=0}^{i-1}(1 - e^{-h_{i'}}), \text{ there is a depth-}h_i \text{ node } v_i$$
$$\text{where the root to } v_i \text{ path has at least } i \text{ random variables of value 1.} \tag{18}$$

For $i = 0$, this follows easily because the root itself is 1 with probability 1. We now assume the induction hypothesis (18) for some $i < c$ and prove it for $i + 1$. 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 19 implies that for any $j \in V_i$, $\mathbb{P}[E'_j] \geq (1 - e^{-h_i})$. Since the events E_j are disjoint, and are independent of $E'_{j'}$ for any $j' \in V_i$, we get

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

Since the events E_j are disjoint, $\sum_j \mathbb{P}[E_j] = \mathbb{P}[\exists j \in V_i : E_j]$. By (18), this probability is at least $\prod_{i'=0}^{i-1}(1 - e^{-h_{i'}})$. Hence,

$$\mathbb{P}[\exists j \in V_i : E_j \wedge E'_j] \geq \prod_{i'=0}^{i} (1 - e^{-h_{i'}}).$$

Note that if events E_j and E'_j are true (for any $j \in V_i$) then there is some depth-h_{i+1} node v_{i+1} with at least $i+1$ r.v.s of value 1 on the root to v_{i+1} path. This proves the inductive statement for $i+1$. Finally, using (18) for $i = c$, the probability that there is a root-leaf path with at least c r.v.s of value one is 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 18 and 20, we obtain:

Theorem 8 *The integrality gap of the LP (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 any group T_j. With probability $1 - (1 - 1/q)^q \approx 1 - 1/e$, there is a task $a_j \in T_j$ for which the random variable X_{a_j} is 1. So the expected number of groups for which this event happens is about $(1 - 1/e)q$. As there is a resource associated with *every* choice of one task from each group, the expected makespan is at least $(1 - 1/e)q$.

Consider the LP relaxation with a target bound of $B = \log q$ 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,

$$\beta_k(X) \leq \frac{1}{\log k} \log\left(1 + \frac{k^{1/\log q}}{q}\right) \leq \frac{2k^{1/\log q}}{q \log k}, \tag{19}$$

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

Lemma 21 *Constraint (8) is satisfied with* $b = 2e^2$ *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 (19), the LHS is at most $kq \cdot \beta_k(X) \leq kq \frac{2q^{1/\log q}}{q \log k} \leq 2e \cdot k$.
- $q < k \leq q^2$. We now use $|L(K)| \leq n = q^2$. By (19), we have $\beta_k(X) \leq \frac{2 \cdot q^{2/\log q}}{q \log k} \leq \frac{2e^2}{q \log k}$. So $LHS \leq q^2 \cdot \beta_k(X) \leq \frac{2e^2 q}{\log k} \leq 2e^2 \cdot k$.
- $k > q^2$. 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 $LHS \leq 2e^2 \cdot k$ in all cases. □

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

We also observe that the integrality gap of LP (2) is $\alpha \leq 2$ for all instances of the deterministic problem that need to be solved in our algorithm. 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. 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 tasks 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 tasks in G_i. The constraints in (2) 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 tasks 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.

6 Conclusion

We considered a class of stochastic makespan minimization problems, where a specific number of tasks need to be selected and each selected task induces a random load on multiple resources. When the set-system (consisting of the tasks and resources) satisfies some geometric properties, we obtained good approximation algorithms. In particular, for stochastic intervals on a line, we obtained an $O(\log \log m)$-approximation algorithm. Our approach was based on a natural LP relaxation, which also has integrality gap $\Omega(\log^* m)$. Finding the correct integrality gap of this LP remains an interesting open question. Obtaining a constant-factor approximation (or hardness results) for stochastic intervals is another interesting direction.

A Scaling the optimal value

Suppose that $\mathcal{A}$ is a polynomial algorithm that given any GenMakespan instance, returns one of the following:

- a solution of objective at most ρ, or
- a certificate that the optimal value is more than 1.

Using this, we provide a polynomial time $O(\rho)$-approximation algorithm for GENMAKESPAN. Observe that the optimal value OPT of GENMAKESPAN lies between $L := \min_{j\in[n]} \mathbb{E}[X_j]$ and $U := n \cdot \max_{j\in[n]} \mathbb{E}[X_j]$. It follows that $\frac{B^*}{2} \le \mathsf{OPT} \le B^*$ for some value B^* in the set:

$$\mathcal{G} := \left\{ 2^\ell \cdot L : 0 \le \ell \le \log_2(U/L) + 1,\ \ell \in \mathbb{Z} \right\}$$

For each $B \in \mathcal{G}$, consider the modified GENMAKESPAN instance with r.v. X_j/B for each task j; and run algorithm $\mathcal{A}$ on this instance. Finally, return the solution with the smallest objective obtained over all $B \in \mathcal{G}$. Note that when $B = B^*$, the optimal value of the modified GENMAKESPAN instance is at most 1: so algorithm $\mathcal{A}$ must find a solution with (modified) objective at most ρ, i.e., the expected makespan under the original r.v.s $\{X_j\}$ is at most $\rho \cdot B^* \le 2\rho \cdot \mathsf{OPT}$. The number of times we call algorithm $\mathcal{A}$ is $O(\log(U/L))$. Note that $L \ge s_{min}$ and $U \le n \cdot s_{max}$ where s_{min} and s_{max} are the minimum and maximum values that the r.v.s take. So, $\log(U/L) = \log(n\frac{s_{max}}{s_{min}})$, which is polynomial in the instance size. Hence, we obtain a polynomial time 2ρ-approximation algorithm for GENMAKESPAN.

B The α-packable property for rectangles and fat objects

In this section, we relate the α-packable property of a set system to the the intergrality gap of the natural LP relaxation for maximum (weighted) independent set for the set system. Using known integrality gap results for maximum independent set for axis-parallel rectangles and fat objects, we can show α-packability of the corresponding set systems for suitable values of α.

Recall the setting in the *α-packable* property. There is a set system $([n], \mathcal{L})$ with size $s_j \ge 0$ and reward r_j for each element $j \in [n]$, and a bound $\theta \ge \max_j s_j$. We are interested in the integrality gap (and a polynomial-time rounding algorithm) for LP (2), restated below.

$$\max \left\{ \sum_{j\in[n]} r_j \cdot y_j \ :\ \sum_{j\in L} s_j \cdot y_j \le \theta,\ \forall L \in \mathcal{L};\ 0 \le y_j \le 1,\ \forall j \in [n] \right\}.$$

When all sizes $s_j = 1$ and the bound $\theta = 1$, we obtain the *independent set* LP:

$$\max \left\{ \sum_{j\in[n]} r_j \cdot y_j \ :\ \sum_{j\in L} y_j \le 1,\ \forall L \in \mathcal{L};\ 0 \le y_j \le 1,\ \forall j \in [n] \right\}. \tag{20}$$

Note that the corresponding integral problem involves selecting a max-reward subset of *disjoint* elements. (Elements e and f are disjoint if there is no set $L \in \mathcal{L}$ with $e, f \in L$.)

Theorem 9 *Suppose that the independent set LP (20) has integrality gap ρ and an associated polynomial time rounding algorithm. Then, the set-system is $O(\rho \cdot \log\log m)$-packable.*

Proof The proof proceeds in several steps: (i) we first consider the special case when all s_j values are 1, but the parameter θ can be arbitrary, (ii) secondly, when $\theta \gg s_j$ for all j (by more than a $\log m$ factor), we use randomized rounding, and (iii) finally, for the general case, we use a standard bucketing trick to create $O(\log \log m)$ groups, and show that one of the above two steps will work for each of these groups.

We give details of the first step. We show a rounding algorithm for the following LP, and show that its integrality gap is at most 2ρ:

$$\max \left\{ \sum_{j \in [n]} r_j \cdot y_j \ : \ \sum_{j \in L} y_j \leq b, \ \forall L \in \mathcal{L}; \ 0 \leq y_j \leq 1, \ \forall j \in [n] \right\}. \tag{21}$$

Here, we assume that $b \geq 1$ is integer. Note that this is a special case of the LP (2) used in the α-packable condition.

(i) Rounding for the LP (21): We combine the rounding algorithm for the independent set LP relaxation (20) with a greedy strategy to round a feasible solution to the LP (21). Let y be a feasible (fractional) solution to the latter LP. We define $\bar{y} = y/b$, which is a feasible solution to the independent set LP relaxation (20).

We build the solution $T \subseteq [n]$ iteratively; initially $T = \emptyset$. For each iteration $k = 1, \cdots b$, we perform the following steps:

1. Consider the solution $\bar{y}$ *restricted* to $[n] \backslash T$. Since this is a feasible solution to the independent set LP (20), we use the independent set rounding algorithm to obtain an integral solution $S_k \subseteq [n] \backslash T$.
2. Update $T \leftarrow T \cup S_k$.

As $\{S_k\}$ are disjoint subsets, $T = \cup_{k=1}^{b} S_k$ is a feasible integral solution to (21).

We now analyze the reward of the solution T. For any subset $U \subseteq [n]$ let $Y(U) := \sum_{j \in U} r_j \cdot y_j$ be the LP-value restricted to U. Consider the two cases:

– Suppose $Y([n] \backslash T) \geq \frac{1}{2} \cdot Y([n])$ at the end of the algorithm. It follows that $Y([n] \backslash T) \geq \frac{1}{2} \cdot Y([n])$ in each iteration k. Consider the LP solution $\bar{y}$ *restricted* to $[n] \backslash T$ (in iteration k). Since the rounding algorithm for the independent set LP relaxation has integrality gap ρ,

$$r(S_k) \geq \frac{1}{\rho} \sum_{j \in [n] \backslash T} r_j \cdot \bar{y}_j = \frac{1}{\rho b} Y([n] \backslash T) \geq \frac{Y([n])}{2\rho b}.$$

Adding over all b iterations, $r(T) = \sum_{k=1}^{b} r(S_k) \geq \frac{Y([n])}{2\rho}$.

– Suppose $Y([n] \backslash T) < \frac{1}{2} \cdot Y([n])$ at the end of the algorithm. Then,

$$r(T) \geq Y(T) = Y([n]) - Y([n] \backslash T) > \frac{1}{2} \cdot Y([n]).$$

In either case, we obtain that the algorithm's reward $r(T) \geq \frac{1}{2\rho} \cdot Y([n])$. This proves that the integrality gap of (21) is at most 2ρ.

(ii) Randomized Rounding for large θ. Let τ denote $\frac{\theta}{2\log m}$. Consider the special case when $s_j \le \tau$ for all $j \in [n]$. In this case, the LP relaxation (2) is a special case of *packing integer programs* (PIPs), studied in [19]. Theorem 3.7 in [19] implies an $O(m^{1/P})$ integrality gap for the LP (2), where

$$P = \frac{\theta}{\max_{j\in[n]} s_j} \ge \frac{\theta}{\tau} = 2\log m.$$

Therefore, the LP (2) has constant integrality gap in this special case.

(iii) Geometric grouping for the general case. We now consider the LP (2) in the general setting. Let y be a fractional solution to this LP. As define above, $\tau := \frac{\theta}{2\log m}$. We first partition the elements into groups based on their sizes as follows:

$$G_k := \begin{cases} \{j \in [n] : s_j < \tau\} & \text{if } k = 0 \\ \{j \in [n] : 2^{k-1}\tau \le s_j < 2^k\tau\} & \text{if } k \ge 1 \end{cases}.$$

Note that the number of groups is $K = O(\log\log m)$ as $\max_j s_j \le \theta$. We handle each group separately, and pick the maximum reward solution across the K groups.

Consider a group $G_k, k \ge 1$. Consider the fractional solution z defined as:

$$z_j = \begin{cases} y_j/4 & \text{if } j \in G_k \\ 0 & \text{otherwise} \end{cases}$$

We claim that z is a feasible solution to the LP (21) restricted to G_k, and a suitable value of b. Indeed, consider any $L \in \mathcal{L}$. Then,

$$\sum_{j\in L} z_j \le \frac{1}{2^{k-1}\tau} \sum_{j\in G_k\cap L} s_j \cdot z_j = \frac{1}{2^{k+1}\tau} \sum_{j\in G_k\cap L} s_j \cdot y_j \le \frac{\theta}{2^{k+1}\tau} \le \frac{c}{2} \le \lfloor c \rfloor,$$

where we have used the fact that y is a feasible solution to (2), and $c := \theta / \max_{j\in G_k} s_j \ge \max\{1, \frac{\theta}{2^k\tau}\}$. It follows that z is a feasible solution to the LP (21) where $b = \lfloor c \rfloor$. Hence, using the rounding algorithm for (21) mentioned in the first step above, we obtain a solution $V_k \subseteq G_k$ with reward

$$r(V_k) \ge \frac{1}{2\rho} \sum_{j\in G_k} r_j \cdot z_j \ge \frac{1}{8\rho} \sum_{j\in G_k} r_j y_j.$$

Moreover, for each $L \in \mathcal{L}$, we have $|V_k \cap L| \le b$. Hence, for any $L \in \mathcal{L}$,

$$\sum_{j\in V_k\cap L} s_j \le \left(\max_{j\in G_k} s_j\right) \cdot |V_k \cap L| = \frac{\theta}{c} \cdot |V_k \cap L| \le \frac{\theta b}{c} \le \theta.$$

Thus, V_k is a feasible integral solution to (2).

Finally we consider the case $k = 0$, i.e., the group G_0. As argued in the second step above, we obtain a solution $V_0 \subseteq G_0$ with reward $r(V_0) \geq \frac{1}{\sigma} \cdot \sum_{j \in G_0} r_j y_j$ where $\sigma \geq 1$ is constant. It follows that V_0 is an integral solution to (2) as well.

Finally, choosing the best solution from $\{V_k\}$ over all groups, we obtain reward at least

$$\max_k r(V_k) \geq \frac{1}{K} \sum_k r(V_k) \geq \frac{1}{K \cdot \max(8\rho, \sigma)} \sum_k \sum_{j \in G_k} r_j y_j = \frac{1}{K \cdot \max(8\rho, \sigma)} \sum_{j \in [n]} r_j y_j.$$

This proves that the integrality gap of (2) is $O(\rho \log \log m)$. □

We now combine Theorem 9 with known results on maximum weight independent sets for rectangles and fat objects, to prove their α-packable property.

Corollary 5 *The set-system where tasks are n axis-aligned rectangles in the plane and resources are all points in the plane, is $O((\log \log n)^2)$-packable.*

Proof The weighted independent set problem for rectangles in the plane has an LP-based $O(\log \log n)$ approximation [7]. Combined with Theorem 9 and the fact that the number of points m can be ensured to be $poly(n)$ (see Sect. 4.3), we obtain that the set-system is $O((\log \log n)^2)$-packable. □

Corollary 6 *The set-system where tasks are n disks (of arbitrary radii) in the plane and resources are all points in the plane, is $O(\log \log n)$-packable.*

Proof There is an LP-based $O(u(n)/n)$-approximation algorithm for weighted independent set on set-systems where the "union complexity" of n objects is at most $u(n)$ [9]. See the survey [2] for more details on union complexity. The union complexity of disks (of arbitrary radii) is $O(n)$. So there is an LP-based $O(1)$-approximation algorithm for weighted independent set. Combined with Theorem 9 and that $m = poly(n)$, the result follows. □

Corollary 7 *The set-system where tasks are n fat triangles in the plane and resources are all points in the plane, is $O(\log^* n \cdot \log \log n)$-packable.*

Proof The union complexity of fat triangles is $u(n) = O(n \log^* n)$ [3]. Using the result from [9], we obtain an LP-based $O(\log^* n)$-approximation algorithm for the weighted independent set problem. Using Theorem 9 and that $m = poly(n)$, the result follows. □

References

1. Agarwal, P.K., Mustafa, N.H.: Independent set of intersection graphs of convex objects in 2d. Comput. Geom. **34**(2), 83–95 (2006)
2. Agarwal, P.K., Pach, J., Sharir, M.: State of the union of geometric objects. In: Surveys in Discrete and Computational Geometry Twenty Years Later, pp. 9–48 (2008)
3. Aronov, B., de Berg, M., Ezra, E., Sharir, M.: Improved bounds for the union of locally fat objects in the plane. SIAM J. Comput. **43**(2), 543–572 (2014)

4. Carr, R.D., Vempala, S.S.: Randomized metarounding. Random Struct. Algorithms **20**(3), 343–352 (2002)
5. Chakrabarti, A., Chekuri, C., Gupta, A., Kumar, A.: Approximation algorithms for the unsplittable flow problem. Algorithmica **47**(1), 53–78 (2007)
6. Chalermsook, P., Chuzhoy, J.: Maximum independent set of rectangles. In: SODA, pp. 892–901 (2009)
7. Chalermsook, P., Walczak, B.: Coloring and maximum weight independent set of rectangles. In: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, pp. 860–868 (2021)
8. Chan, T.M.: A note on maximum independent sets in rectangle intersection graphs. Inf. Process. Lett. **89**(1), 19–23 (2004)
9. Chan, T.M., Har-Peled, S.: Approximation algorithms for maximum independent set of pseudo-disks. Discrete Comput. Geom. **48**(2), 373–392 (2012)
10. Chekuri, C., Mydlarz, M., Shepherd, F.B.: Multicommodity demand flow in a tree and packing integer programs. ACM Trans. Algorithms **3**(3), 27 (2007)
11. Chekuri, C., Vondrák, J., Zenklusen, R.: Dependent randomized rounding via exchange properties of combinatorial structures. In: 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23–26, 2010, Las Vegas, Nevada, USA, pp. 575–584 (2010)
12. Cornuejols, G., Fisher, M.L., Nemhauser, G.L.: Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. Manag. Sci. **23**(8), 789–810 (1977)
13. 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)
14. Gupta, A., Kumar, A., Nagarajan, V., Shen, X.: Stochastic load balancing on unrelated machines. Math. Oper. Res. **46**(1), 115–133 (2021)
15. Hui, J.Y.: Resource allocation for broadband networks. IEEE J. Sel. Areas Commun. **6**(3), 1598–1608 (1988)
16. Kelly, F.P.: Notes on effective bandwidths. In: Stochastic Networks: Theory and Applications, pp. 141–168. Oxford University Press (1996)
17. Kleinberg, J., Rabani, Y., Tardos, E.: Allocating bandwidth for bursty connections. SIAM J. Comput. **30**(1), 191–217 (2000)
18. Molinaro, M.: Stochastic ℓ_p load balancing and moment problems via the l-function method. In: SODA, pp. 343–354 (2019)
19. Srinivasan, A.: Improved approximation guarantees for packing and covering integer programs. SIAM J. Comput. **29**(2), 648–670 (1999)