Adaptivity and Approximation for Stochastic Packing Problems*

Brian C. Dean† Michel X. Goemans‡ Jan Vondrák§

Abstract

We study stochastic variants of Packing Integer Programs (PIP) — the problems of finding a maximum-value 0/1 vector x satisfying $Ax \leq b$, with A and b nonnegative. Many combinatorial problems belong to this broad class, including the knapsack problem, maximum clique, stable set, matching, hypergraph matching (a.k.a. set packing), b-matching, and others. PIP can also be seen as a "multi-dimensional" knapsack problem where we wish to pack a maximum-value collection of items with vector-valued sizes. In our stochastic setting, the vector-valued size of each item is known to us apriori only as a probability distribution, and the size of an item is instantiated once we commit to including the item in our solution.

Following the framework of [3], we consider both adaptive and non-adaptive policies for solving such problems, adaptive policies having the flexibility of being able to make decisions based on the instantiated sizes of items already included in the solution. We investigate the *adaptivity gap* for these problems: the maximum ratio between the expected values achieved by optimal adaptive and non-adaptive policies. We show tight bounds on the adaptivity gap for set packing and b-matching, and we also show how to find efficiently non-adaptive policies approximating the adaptive optimum. For instance, we can approximate the adaptive optimum for stochastic set packing to within $O(d^{1/2})$, which is not only optimal with respect to the adaptivity gap, but it is also the best known approximation factor in the deterministic case. It is known that there is no polynomial-time $d^{1/2-\epsilon}$-approximation for set packing, unless $NP = ZPP$. Similarly, for b-matching, we obtain algorithmically a tight bound on the adaptivity gap of $O(\lambda)$ where λ satisfies $\sum \lambda^{b_j+1} = 1$.

For general Stochastic Packing, we prove that a simple greedy algorithm provides an $O(d)$-approximation to the adaptive optimum. For $A \in [0,1]^{d\times n}$, we provide an $O(\lambda)$-approximation where $\sum 1/\lambda^{b_j} = 1$. (For $b = (B, B, \ldots, B)$, we get $\lambda = d^{1/B}$.) We also improve the hardness results for deterministic PIP: in the general case, we prove that a polynomial-time $d^{1-\epsilon}$-approximation algorithm would imply $NP = ZPP$. In the special case when $A \in [0,1]^{d\times n}$ and $b = (B, B, \ldots, B)$, we show that a $d^{1/B-\epsilon}$-approximation would imply $NP = ZPP$. Finally, we prove that it is PSPACE-hard to find the optimal adaptive policy for Stochastic Packing in any fixed dimension $d \geq 2$.

*Research supported in part by NSF contracts ITR-0121495 and CCR-0098018.

†M.I.T., CSAIL, Cambridge, MA 02139. bdean@mit.edu

‡M.I.T., Dept. of Math. and CSAIL, Cambridge, MA 02139. goemans@math.mit.edu

§M.I.T., Dept. of Math., Cambridge, MA 02139. vondrak@math.mit.edu

1 Stochastic Packing

We consider a multidimensional generalization of the *Stochastic Knapsack* problem [3] where instead of a scalar size, each item has a "vector size" in $\mathbb{R}^d$, and a feasible solution is a set of items such that the total size is bounded by a given capacity in each component. This problem can be also seen as the stochastic version of a *Packing Integer Program* (PIP), defined in [10]. A Packing Integer Program is a combinatorial problem in a very general form involving the computation of a solution $x \in \{0,1\}^d$ satisfying *packing constraints* of the form $Ax \leq b$ where A is nonnegative. This encapsulates many combinatorial problems such as hypergraph matching (a.k.a. set packing), b-matching, disjoint paths in graphs, maximum clique and stable set. In general, Packing Integer Programs are NP-hard to solve or even to approximate well. We mention the known hardness results in Section 1.1.

Our stochastic generalization follows the philosophy of [3] where items have independent random sizes which are determined and revealed to our algorithm only after an item is chosen to be included in the solution. Before the algorithm decides to insert an item, it only has some information about the probability distribution of its size. In the PIP problem, the "size" of an item is a column of the matrix A, which we now consider to be a random vector independent of other columns of A. Once an item is chosen, the size vector is fixed, and the item cannot be removed from the solution anymore. An algorithm whose decisions depend on the observed size vectors is called *adaptive*; an algorithm which chooses an entire sequence of items in advance is *non-adaptive*.

Definition 1.1. (PIP) *Given a matrix $A \in \mathbb{R}_+^{d\times n}$ and vectors $\vec{b} \in \mathbb{R}_+^d, \vec{v} \in \mathbb{R}_+^n$, a Packing Integer Program (PIP) is the problem of maximizing $\vec{v} \cdot \vec{x}$ subject to $A\vec{x} \leq \vec{b}$ and $\vec{x} \in \{0,1\}^d$.*

Definition 1.2. (Stochastic Packing) *Stochastic Packing (SP) is a stochastic variant of a PIP where A is a random matrix whose columns are independent*

random vectors, denoted $\vec{S}(1), \ldots \vec{S}(n)$. *A feasible solution is a set of items* F *such that* $\sum_{i \in F} \vec{S}(i) \leq \vec{b}$. *The value of* $\vec{S}(i)$ *is instantiated and fixed once we include item* i *in* F. *Once this decision is made, the item cannot be removed. Whenever the condition* $\sum_{i \in F} \vec{S}(i) \leq \vec{b}$ *is violated, no further items can be inserted, and no value is received for the overflowing item.*

We consider 4 classes of Stochastic Packing problems: (1) General Stochastic Packing where no restrictions are placed on item sizes or capacity; by scaling, we can assume that $\vec{b} = (1, 1, \ldots, 1)$. *(2) Restricted Stochastic Packing where* $\vec{S}(i)$ *have values in* $[0,1]^d$ *and* $\vec{b} \in \mathbb{R}^d, b_j \geq 1$. *(3) Stochastic Set Packing where* $\vec{S}(i)$ *have values in* $\{0,1\}^d$ *and* $\vec{b} = (1, 1, \ldots, 1)$. *(4) Stochastic b-matching where* $\vec{S}(i)$ *have values in* $\{0,1\}^d$ *and* $\vec{b} \in \mathbb{Z}^d, b_j \geq 1$.

Definition 1.3. *An adaptive policy for a Stochastic Packing problem is a function* $\mathcal{P} : 2^{[n]} \times \mathbb{R}_+^d \rightarrow [n]$. *The interpretation of* $\mathcal{P}$ *is that given a configuration* $(A, \vec{b})$ *where* A *represents the items still available and* $\vec{b}$ *the remaining capacity,* $\mathcal{P}(A, \vec{b})$ *determines which item should be chosen next among the items in* A.

A non-adaptive policy is a fixed ordering of items to be inserted.

For an instance of Stochastic Packing, let ADAPT denote the maximum possible expected value achieved by an adaptive policy, and NONADAPT the maximum possible expected value achieved by a non-adaptive policy.

Given a Stochastic Packing problem, we can ask several questions. The first one is, what is the optimum adaptive policy? Can we find or characterize the optimum adaptive policy? Next, we can ask, what is the *benefit of adaptivity* and evaluate the *adaptivity gap* – the ratio between the expected values of optimal adaptive and non-adaptive policies? Note that such a question is independent of any complexity-theoretic assumptions; it refers merely to the existence of policies, which may not be computable efficiently. We can also try to find algorithmically a good non-adaptive policy, which approximates the optimal *adaptive policy* within some factor. In the single-dimensional case, we proved that the adaptivity gap is bounded by a constant factor, and the non-adaptive solution can be achieved by a simple greedy algorithm [3].

1.1 Known results. Stochastic Packing in this form has not been considered before. We build on the previous work on Packing Integer Programs, which was initiated by Raghavan and Thompson [10], [9]. They proposed an LP approach combined with randomized rounding, which yields an $O(d)$-approximation for the general case [10]. For the case of Set Packing, their methods yield an $O(\sqrt{d})$-approximation. For general $\vec{b}$ parametrized by $B = \min b_i$, there is an $O(d^{1/B})$-approximation for $A \in [0,1]^{d \times n}$ and an $O(d^{1/(B+1)})$-approximation for $A \in \{0,1\}^{d \times n}$. The greedy algorithm gives a $\sqrt{d}$-approximation for Set Packing [5].

This is complemented by the hardness results of Chekuri and Khanna [1] who show that a $d^{1/(B+1)-\epsilon}$-approximation for the b-matching problem with $\vec{b} = (B, B, \ldots, B)$ would imply $NP = ZPP$ (using Håstad's result on the inapproximability of Max Clique [6]). For $A \in [0,1]^{d \times n}$ and real $B \geq 1$, they get hardness of $d^{1/(\lfloor B \rfloor + 1)-\epsilon}$-approximation; this still leaves a gap between $O(d^{1/B})$ and $d^{1/(B+1)-\epsilon}$ for B integer, in particular a gap between $d^{1/2-\epsilon}$ and $O(d)$ in the general case.

The analysis of the randomized rounding technique has been refined by Srinivasan [11] who presents stronger bounds; however, the approximation factors are not improved in general (and this is essentially impossible due to [1]).

1.2 Our results. We prove bounds on the adaptivity gap and we also present algorithms to find a good non-adaptive solution. Our results are summarized in the table below. In the table, GP = General Packing, SP = Set Packing, RP = Restricted Packing, BM = b-matching and previously known results are within square brackets.

	Determ. approx.	Inapprox-imability	Adaptivity gap	Stochastic approx.
GP	$[\ O(d)\ ]$	$d^{1-\epsilon}$	$\Omega(\sqrt{d})$	$O(d)$
RP	$[\, O\left(d^{\frac{1}{B}}\right)]$	$d^{\frac{1}{B}-\epsilon}$	$\Omega\left(d^{\frac{1}{B+1}}\right)$	$O\left(d^{\frac{1}{B}}\right)$
SP	$[\, O\left(\sqrt{d}\right)]$	$[\, d^{1/2-\epsilon}\,]$	$\Theta(\sqrt{d})$	$O(\sqrt{d})$
BM	$[O\left(d^{\frac{1}{B+1}}\right)]$	$[\, d^{\frac{1}{B+1}-\epsilon}\,]$	$\Theta\left(d^{\frac{1}{B+1}}\right)$	$O\left(d^{\frac{1}{B+1}}\right)$

It turns out that the adaptivity gap for Stochastic Set Packing can be $\Omega(\sqrt{d})$, and in the case of b-matching it can be $\Omega(\lambda)$ where $\lambda \geq 1$ is the solution of $\sum_j 1/\lambda^{b_j+1} = 1$. For $\vec{b} = (B, B, \ldots, B)$, we get $\lambda = d^{1/(B+1)}$. (It is quite conspicuous how these bounds match the inapproximability of the deterministic problems, which is seemingly an unrelated notion!) These instances are described in Section 2.

On the positive side, we prove in Section 4 that a natural extension of the greedy algorithm of [3] finds a non-adaptive solution of expected value $ADAPT/O(d)$ for the general case. For Stochastic Set Packing we can achieve non-adaptively expected value $ADAPT/O(\sqrt{d})$

by an LP approach with randomized rounding. The LP which is described in Section 3 provides a non-trivial upper bound on the expected value achieved by any adaptive strategy, and our randomized rounding strategy is described in Section 5. More generally, for stochastic b-matching we can achieve non-adaptively $ADAPT/O(\lambda)$ where $\sum_j 1/\lambda^{b_j+1} = 1$. For Restriced Stochastic Packing we get $ADAPT/O(\lambda)$ where $\sum_j 1/\lambda^{b_j} = 1$ (Section 7). Note that for $\vec{b} = (B, B, \ldots, B)$, we get $\lambda = d^{1/(B+1)}$ for Stochastic b-matching and $\lambda = d^{1/B}$ for Restricted Stochastic Packing, i.e. the best approximation factors known in the deterministic case.

In Section 8, we improve the hardness results for deterministic PIP: We prove that a polynomial-time $d^{1-\epsilon}$-approximation algorithm would imply $NP = ZPP$, so our greedy algorithm is essentially optimal even in the deterministic case. We also improve the hardness result to $d^{1/B-\epsilon}$ in the case when $A \in [0,1]^{d\times n}, \vec{b} = (B, B, \ldots, B)$, $B \geq 2$ integer.

All our approximation factors match the best results known in the deterministic case and the hardness results imply that these are essentially optimal. For Set Packing and b-matching, we also match our lower bounds on the adaptivity gap. It is quite surprising that we can approximate the optimal *adaptive* policy to within $O(\lambda)$ efficiently, while this can be the actual gap between the adaptive and non-adaptive policies; and unless $NP = ZPP$, we cannot approximate even the best *non-adaptive* solution better than this!

Our results only assume that we know the expected size of each item (or more precisely the truncated mean size, see Definition 3.1, and the probability that an item alone fits) rather than the entire probability distribution. With such limited information, our results are also tight in the following sense. For Stochastic Packing and Stochastic Set Packing, there exist instances with the same mean item sizes for which the optimum adaptive values differ by a factor $\Theta(d)$ or $\Theta(\sqrt{d})$, resp. These instances are described in Section 6.

2 The benefit of adaptivity

We present examples which demonstrate that for Stochastic Packing, adaptivity can bring a substantial advantage (as opposed to Stochastic Knapsack where the benefit of adaptivity is only a constant factor [3]). Our examples are simple instances of Set Packing and b-matching ($A \in \{0,1\}^{d\times n}$).

LEMMA 2.1. *There are instances of Stochastic Set Packing, such that*

$$ADAPT \geq \frac{\sqrt{d}}{2}\ NONADAPT.$$

Proof. Define items of type $i = 1, 2, \ldots, d$ where items of type i have size vector $\vec{S}(i) = Be(p)\ \vec{e}_i$, i.e. a random Bernoulli variable $Be(p)$ in the i-th component, and 0 in the remaining components ($p > 0$ to be chosen later). We have an unlimited supply of items of each type. All items have unit value and we assume unit capacity $\vec{b} = (1, 1, \ldots 1)$.

An adaptive policy can insert items of each type until a size of 1 is attained in the respective component; the expected number of items of each type inserted is $1/p$. Therefore $ADAPT \geq d/p$.

On the other hand, consider a set of items F. We estimate the probability that F is a feasible solution. For every component i, let k_i denote the number of items of type i in F. We have:

$$\begin{aligned}\Pr[S_i(F) \leq 1] &= (1-p)^{k_i} + k_i p(1-p)^{k_i-1} \\ &= (1 + p(k_i-1))(1-p)^{k_i-1} \\ &\leq (1+p)^{k_i-1}(1-p)^{k_i-1} \\ &= (1-p^2)^{k_i-1}\end{aligned}$$

and

$$\begin{aligned}\Pr[||\vec{S}(F)||_\infty \leq 1] &\leq \prod_{i=1}^{d}(1-p^2)^{k_i-1} \\ \leq e^{-p^2\sum_{i=1}^{d}(k_i-1)} &= e^{-p^2(|F|-d)}.\end{aligned}$$

Thus the probability that a set of items fits decreases exponentially with its size. For any non-adaptive policy, the probability that the first k items in the sequence are inserted successfully is at most $e^{-p^2(k-d)}$, and we can estimate the expected value achieved.

$$\begin{aligned}NONADAPT &= \sum_{k=1}^{\infty}\Pr[k \text{ items fit}] \\ \leq d + \sum_{k=d+1}^{\infty} e^{-p^2(k-d)} &= d + \frac{1}{e^{p^2}-1} \leq d + \frac{1}{p^2}.\end{aligned}$$

We choose $p = d^{-1/2}$ for which we get $ADAPT \geq d^{3/2}$ and $NONADAPT \leq 2d$.

We generalize this example to an arbitrary integer capacity vector $\vec{b}$.

LEMMA 2.2. *There are instances of Stochastic b-matching such that*

$$ADAPT \geq \frac{\lambda}{4}\ NONADAPT$$

where $\lambda \geq 1$ is the solution of $\sum_{i=1}^{d} 1/\lambda^{b_i+1} = 1$.

Proof. Let $p \leq 1$ satisfy $\sum_{i=1}^{d} p^{b_i+1} = 1$. Consider the same set of items that we used in the previous proof,

only the values are modified as $v_i = p^{b_i+1}/(b_i+1)$. The same adaptive strategy will now insert items of each type, until it accumulates size b_i in the i-th component. The expected number of items of type i inserted will be b_i/p, and therefore

$$ADAPT \geq \sum_{i=1}^{d} v_i \frac{b_i}{p} = \frac{1}{p} \sum_{i=1}^{d} \frac{b_i p^{b_i+1}}{b_i+1} \geq \frac{1}{2p}.$$

Consider a set of items F. We divide the items of each type into blocks of size b_i+1 (for type i). Suppose that the number of blocks of type i is k_i. We estimate the probability that F is a feasible solution; we use the fact that each block alone has a probability of overflow p^{b_i+1}, and these events are independent:

$$\begin{aligned} \Pr[S_i(F) \leq b_i] &\leq (1-p^{b_i+1})^{k_i} \leq e^{-k_i p^{b_i+1}}, \\ \Pr[\vec{S}(F) \leq \vec{b}] &\leq e^{-\sum k_i p^{b_i+1}}. \end{aligned}$$

Now we express this probability as a function of the value of F. We defined the blocks in such a way that a block of type i gets value p^{b_i+1}, and $\sum k_i p^{b_i+1}$ is the value of all the blocks. There might be items of value less than p^{b_i+1} of type i, which are not assigned to any block. All these together can have value at most 1 (by the definition of p). So the probability that the non-adaptive policy fits a set of value at least w is $\Psi(w) \leq \min\{e^{1-w}, 1\}$. Now we can estimate the expected value achieved by any non-adaptive policy:

$$NONADAPT = \int_0^\infty \Psi(w)dw \leq 1 + \int_1^\infty e^{1-w}dw = 2.$$

It follows that the adaptivity gap is at least $1/4p = \lambda/4$ where λ satisfies $\sum_{i=1}^{d} 1/\lambda^{b_i+1} = 1$.

As a special case, for $\vec{b} = (B, B, \ldots, B)$, the lemma holds with $\lambda = d^{1/(B+1)}$, which is the adaptivity gap for stochastic b-matching that we claimed. In Section 5, we prove that for Set Packing and b-matching, these bounds are not only tight, but they can be actually achieved by a polynomial-time non-adaptive policy.

On the other hand, the best lower bound we have on the adaptivity gap in the general case is $\Omega(\sqrt{d})$ (from Set Packing), and we do not know whether this is the largest possible gap. Our best upper bound is $O(d)$, as implied by the greedy approximation algorithm (Section 4).

3 Bounding an adaptive policy

In this section, we introduce a linear program which allows us to upper bound the expected value of any adaptive policy. This is based on the same tools that we used in [3] for the 1-dimensional case. This LP together with randomized rounding will be used in Section 5 to design good non-adaptive policies.

DEFINITION 3.1. *For an item with random size vector $\vec{S}$, we define the truncated mean size $\vec{\mu}$ by components as*

$$\mu_j = \mathbf{E}[\min\{S_j, 1\}].$$

For a set of items A, we write $\vec{\mu}(A) = \sum_{i \in A} \vec{\mu}(i)$.

The following lemma can be proved using the same martingale argument that we used in [3]. Here, we show an alternative elementary proof.

LEMMA 3.1. *For any adaptive policy, let A denote the (random) set of items that the policy attempts to insert. Then for each component j, $\mathbf{E}[\mu_j(A)] \leq b_j + 1$ where $\vec{b}$ is the capacity vector.*

Proof. Consider component j. Denote by $M(c)$ the maximum expected $\mu_j(A)$ for a set A that an adaptive policy can possibly try to insert within capacity c in the j-th component. (For now, all other components can be ignored.) We prove, by induction on the number of available items, that $M(c) \leq c+1$.

Suppose that an optimal adaptive policy, given remaining capacity c, inserts item i. Denote by $fit(i,c)$ the characteristic function of the event that item i fits ($S_j(i) \leq c$) and by $s(i)$ its truncated size ($s(i) = \min\{S_j(i), 1\}$). We have

$$\begin{aligned} M(c) &\leq \mu_j(i) + \mathbf{E}[fit(i,c)M(c-s(i))] \\ &= \mathbf{E}[s(i) + fit(i,c)M(c-s(i))] \end{aligned}$$

and using the induction hypothesis,

$$\begin{aligned} M(c) &\leq \mathbf{E}[s(i) + fit(i,c)(c-s(i)+1)] \\ &= \mathbf{E}[fit(i,c)(c+1) + (1-fit(i,c))s(i)] \leq c+1, \end{aligned}$$

completing the proof.

We can bound the value achieved by any adaptive policy using a linear program. Even though an adaptive policy can make decisions based on the observed sizes of items, the total probability that an item i is inserted by the policy is determined beforehand — if we think of a policy in terms of a decision tree, this total probability is obtained by averaging over all the branches of the decision tree where item i is inserted, weighted by the probabilities of executing those branches (which are determined by the policy and distributions of item sizes). We do not actually need to write out this probability explicitly in terms of the policy. Just denote by x_i the total probability that the policy tries to insert item i.

DEFINITION 3.2. *We define the "effective value" of each item i as $w_i = v_i \Pr[\text{item } i \text{ alone fits}]$.*

Conditioned on item i being inserted, the expected value for it is at most w_i. Therefore, the expected value achieved by the policy is at most $\sum_i x_i w_i$. The expected size inserted is $\mathbf{E}[\vec{\mu}(A)] = \sum_i x_i \vec{\mu}(i)$. We know that for any adaptive policy this is bounded by $b_j + 1$ in the j-th component, so we can write the following LP:

$$\mathcal{V} = \max \left\{ \sum_i x_i w_i : \begin{array}{ll} \sum_i x_i \mu_j(i) \le b_j + 1 & \forall j \\ 0 \le x_i \le 1 & \forall i \end{array} \right\}.$$

In this extended abstract, we will be using this LP only for our special cases in which an item always fits when placed alone, i.e. $w_i = v_i$.

Note the similarity between this LP and the usual linear relaxation of the deterministic packing problem. The only difference is that we have $b_j + 1$ instead of b_j on the right-hand side, and yet this LP bounds the performance of any adaptive policy — as we have seen in Section 2, a much more powerful paradigm in general. We will put this linear program to use in Section 5. We summarize:

LEMMA 3.2. *$ADAPT \le \mathcal{V}$.*

4 The greedy algorithm

A straightforward generalization of the greedy algorithm from [3] gives an $O(d)$-approximation algorithm for General Stochastic Packing. Let's go briefly over the main points of the analysis.

Remember that, in the general case, we can assume by scaling that $\vec{b} = (1, 1, \ldots, 1)$. Then a natural measure of multidimensional size is the l_1 norm of the mean size vector:

$$||\vec{\mu}(A)||_1 = \sum_{j=1}^{d} \mu_j(A).$$

The reason to use the l_1 norm here is that it bounds the probability that a set of items overflows. Also, the l_1 norm is easy to work with, because it's linear and additive for collections of items.

LEMMA 4.1. $\Pr[||\vec{S}(A)||_\infty \le 1] \ge 1 - ||\vec{\mu}(A)||_1$.

Proof. For each component, Markov's inequality gives us $\Pr[S_j(A) \ge 1] = \Pr[\min\{S_j(A), 1\} \ge 1] \le \mathbf{E}[\min\{S_j(A), 1\}] \le \mu_j(A)$, and by the union bound $\Pr[||\vec{S}(A)||_\infty \ge 1] \le \sum_{j=1}^{d} \mu_j(A) = ||\vec{\mu}(A)||_1$.

We set a threshold $\sigma \in (0, 1)$ and we define heavy items to be those with $||\vec{\mu}(i)||_1 > \sigma$ and light items those with $||\vec{\mu}(i)||_1 \le \sigma$.

The greedy algorithm.
Take the more profitable of the following:

- A single item, achieving
$$m_1 = \max_i v_i \Pr[||\vec{S}(i)||_\infty \le 1].$$
- A sequence of light items, in the order of decreasing $v_i / ||\vec{\mu}(i)||_1$. This achieves expected value at least
$$m_G = \sum_{k=1}^{n^*} v_k (1 - M_k)$$
where $M_k = \sum_{i=1}^{k} ||\vec{\mu}(i)||_1$, and $n^* = \max\{k : M_k < 1\}$.

We employ the following two lemmas from [3], in which we only replace μ by $||\vec{\mu}||_1$ (which works thanks to Lemma 4.1). As in [3], we set $\sigma = 1/3$.

LEMMA 4.2. *For $\sigma = 1/3$, the expected value an adaptive policy gets for heavy items is*

$$\mathbf{E}[v(H)] \le \mathbf{E}[|H'|]\ m_1 \le 3\mathbf{E}[||\vec{\mu}(H')||_1]\ m_1$$

where H' is the set of heavy items the policy attempts to insert.

LEMMA 4.3. *For $\sigma = 1/3$, the expected value an adaptive policy gets for light items is*

$$\mathbf{E}[v(L)] \le (1 + 3\mathbf{E}[||\vec{\mu}(L)||_1])\ m_G.$$

We observe that for the random set A' that an adaptive policy tries to insert, Lemma 3.1 implies

$$\mathbf{E}[||\vec{\mu}(A')||_1] = \sum_{j=1}^{d} \mathbf{E}[\mu_j(A')] \le 2d.$$

Therefore $\mathbf{E}[||\vec{\mu}(H')||_1] + \mathbf{E}[||\vec{\mu}(L)||_1] \le 2d$ and we get the following.

THEOREM 4.1. *The greedy algorithm for Stochastic Packing achieves expected value at least $GREEDY = \max\{m_1, m_G\}$, and*

$$ADAPT \le (1 + 6d)\ GREEDY.$$

This also proves that the adaptivity gap for Stochastic Packing is at most $O(d)$. It remains an open question whether the gap can actually be $\Theta(d)$. Our best lower bound is $\Omega(\sqrt{d})$, see Section 2.

5 Stochastic Set Packing and b-matching

As a special case, consider the Stochastic Set Packing problem. We have seen that in this case the adaptivity gap can be as large as $\Theta(\sqrt{d})$. We prove that this is indeed tight. Moreover, we present an algorithmic

approach to find an $O(\sqrt{d})$-approximate non-adaptive policy.

Our solution will be a fixed collection of items — that is, we insert all these items, and we collect a nonzero profit only if all the respective sets turn out to be disjoint. The first step is to replace the l_1 norm by a stronger measure of size, which allows one to estimate better the probability that a collection of items is a feasible solution.

DEFINITION 5.1. *For a set of items A,*

$$\hat{\mu}(A) = \sum_{\{i,j\} \in \binom{A}{2}} \vec{\mu}(i) \cdot \vec{\mu}(j).$$

LEMMA 5.1. *For a set of items A with size vectors in $\{0,1\}^d$,*

$$\Pr[||\vec{S}(A)||_\infty \leq 1] \geq 1 - \hat{\mu}(A).$$

Proof. A set of items can overflow in coordinate l only if at least two items attain size 1 in that coordinate. For a pair of items $\{i,j\}$, the probability of this happening is $\mu_l(i)\mu_l(j)$. By the union bound:

$$\begin{aligned} Pr[S_l(A) > 1] &\leq \sum_{\{i,j\} \in \binom{A}{2}} \mu_l(i)\mu_l(j), \\ Pr[||\vec{S}(A)||_\infty > 1] &\leq \sum_{\{i,j\} \in \binom{A}{2}} \vec{\mu}(i) \cdot \vec{\mu}(j) = \hat{\mu}(A). \end{aligned}$$

Now we use the LP formulation introduced in Section 3. Since we can solve the LP in polynomial time, we can assume that we have a solution $\vec{x}$ such that $||\sum x_i \vec{\mu}(i)||_\infty \leq 2$, and $\mathcal{V} = \sum x_i w_i = \sum x_i v_i$ bounds the expected value of any adaptive policy (Lemma 3.2). We can also assume that the value of any item is at most $\frac{\beta}{\sqrt{d}}\mathcal{V}$ for some fixed $\beta > 0$, otherwise the most valuable item alone is a $\frac{\sqrt{d}}{\beta}$-approximation of the optimum.

We sample a random set of items F, item i independently with probability $q_i = \frac{\alpha}{\sqrt{d}} x_i$. Constants α, β will be chosen later. We estimate the expected value that we get for the set obtained in this way. Note that there are "two levels of expectation" here: one related to our sampling, and another to the resulting set being used as a solution of a stochastic problem. The expectation denoted by $\mathbf{E}[]$ in the following computation is the one related to our sampling. Using Lemma 5.1, we can lower bound the expected value obtained by inserting set F by $v(F)(1 - \hat{\mu}(F))$. The expectation of this value with respect to our random sampling is:

$$\begin{aligned} &\mathbf{E}[v(F)(1-\hat{\mu}(F))] \\ &= \mathbf{E}\left[\sum_{i \in F} v_i - \sum_{i \in F} v_i \sum_{\{j,k\} \in \binom{F}{2}} \vec{\mu}(j) \cdot \vec{\mu}(k)\right] \\ &= \mathbf{E}\left[\sum_{i \in F} v_i\right] - \mathbf{E}\left[\sum_{\{j,k\} \in \binom{F}{2}} (v_j + v_k)\vec{\mu}(j) \cdot \vec{\mu}(k)\right] \\ &\quad -\mathbf{E}\left[\sum_{\{j,k\} \in \binom{F}{2}} \sum_{i \in F \setminus \{j,k\}} v_i \vec{\mu}(j) \cdot \vec{\mu}(k)\right] \\ &\geq \sum_i q_i v_i - \sum_{j,k} q_j q_k v_j \vec{\mu}(j) \cdot \vec{\mu}(k) \\ &\quad -\frac{1}{2}\sum_{i,j,k} q_i q_j q_k v_i \vec{\mu}(j) \cdot \vec{\mu}(k) \\ &\geq \frac{\alpha}{\sqrt{d}}\mathcal{V} - \frac{\alpha^2\beta}{d^{3/2}}\mathcal{V}\left(\sum_j x_j \vec{\mu}(j)\right) \cdot \left(\sum_k x_k \vec{\mu}(k)\right) \\ &\quad -\frac{\alpha^3}{2d^{3/2}}\mathcal{V}\left(\sum_j x_j \vec{\mu}(j)\right) \cdot \left(\sum_k x_k \vec{\mu}(k)\right) \\ &\geq \frac{\alpha}{\sqrt{d}}(1 - 4\alpha\beta - 2\alpha^2)\mathcal{V}, \end{aligned}$$

where we used $v_j \leq \frac{\beta}{\sqrt{d}}\mathcal{V}$ and $||\sum x_j \vec{\mu}(j)||_\infty \leq 2$.

We choose α and β to satisfy $\alpha(1 - 4\alpha\beta - 2\alpha^2) = \beta$ and then maximize this value, which yields $\alpha^2 = (-5 + \sqrt{33})/8$ and $\beta^2 = (11\sqrt{33} - 59)/128$. Then $\sqrt{d}/\beta < 5.6\sqrt{d}$ is our approximation factor. Using the method of conditional expectations (on $\mathbf{E}[v(F)(1-\hat{\mu}(F))]$ which can be computed exactly), we can also find the set F in a deterministic fashion. We summarize:

THEOREM 5.1. *For Stochastic Set Packing, there is a polynomial-time algorithm which finds a set of items yielding expected value at least $ADAPT/5.6\sqrt{d}$. Therefore $ADAPT \leq 5.6\sqrt{d}\ NONADAPT$.*

This closes the adaptivity gap for the Stochastic Set Packing problem up to a constant factor, since we know from Section 2 that it could be as large as $\frac{1}{2}\sqrt{d}$.

Next, we sketch how this algorithm generalizes to b-matching, with an arbitrary integer vector $\vec{b}$. A natural generalization of $\hat{\mu}(A)$ and Lemma 5.1 is the following.

DEFINITION 5.2. *For a set of items A,*

$$\hat{\mu}_{\vec{b}}(A) = \sum_{l=1}^{d} \sum_{B \in \binom{A}{b_l+1}} \prod_{i \in B} \mu_l(i).$$

LEMMA 5.2. *For a set of items A with size vectors in $\{0,1\}^d$,*

$$\Pr[\vec{S}(A) \leq \vec{b}] \geq 1 - \hat{\mu}_{\vec{b}}(A).$$

Proof. Similarly to Lemma 5.1, a set of items can overflow in coordinate l, only if $b_l + 1$ items attain size 1

in their j-th component. This happens with probability $\prod_{i\in B}\mu_l(i)$ and we apply the union bound.

Using this measure of size, we apply a procedure similar to our solution of Stochastic Set Packing. We solve

$$\mathcal{V} = \max\left\{\sum_i x_i v_i : \begin{array}{ll}\sum_i x_i\mu_l(i) \le b_l+1 & \forall l\\ 0\le x_i\le 1 & \forall i\end{array}\right\}$$

which is an upper bound on the optimum by Lemma 3.2. We assume that the value of each item is at most $\frac{\beta}{\lambda}\mathcal{V}$ and we sample F, each item i with probability $q_i = \frac{\alpha}{\lambda}x_i$ where $\sum_j 1/\lambda^{b_j+1} = 1$; $\alpha,\beta > 0$ to be chosen later. We estimate the expected value of F:

$$\begin{aligned}
&\mathbf{E}[v(F)(1-\mu_{\vec{b}}(F))]\\
&= \mathbf{E}\left[\sum_{i\in F} v_i\right] - \sum_{l=1}^{d}\mathbf{E}\left[\sum_{B\in\binom{F}{b_l+1}}\sum_{i\in B} v_i\prod_{j\in B}\mu_l(j)\right]\\
&\quad - \sum_{l=1}^{d}\mathbf{E}\left[\sum_{B\in\binom{F}{b_l+1}}\sum_{i\in F\setminus B} v_i\prod_{j\in B}\mu_l(j)\right]\\
&\ge \sum_i q_i v_i - \frac{\beta}{\lambda}\mathcal{V}\sum_{l=1}^{d}(b_l+1)\sum_{|B|=b_l+1}\prod_{j\in B} q_j\mu_l(j)\\
&\quad - \sum_{l=1}^{d}\sum_i q_i v_i\sum_{|B|=b_l+1}\prod_{j\in B} q_j\mu_l(j)\\
&\ge \frac{\alpha}{\lambda}\mathcal{V} - \frac{\beta}{\lambda}\mathcal{V}\sum_{l=1}^{d}\frac{1}{b_l!}\left(\sum_i q_i\mu_l(i)\right)^{b_l+1}\\
&\quad - \frac{\alpha}{\lambda}\mathcal{V}\sum_{l=1}^{d}\frac{1}{(b_l+1)!}\left(\sum_i q_i\mu_l(i)\right)^{b_l+1}\\
&\ge \frac{\alpha}{\lambda}\mathcal{V} - \frac{\beta}{\lambda}\mathcal{V}\sum_{l=1}^{d}\frac{1}{b_l!}\left(\frac{\alpha}{\lambda}(b_l+1)\right)^{b_l+1}\\
&\quad - \frac{\alpha}{\lambda}\mathcal{V}\sum_{l=1}^{d}\frac{1}{(b_l+1)!}\left(\frac{\alpha}{\lambda}(b_l+1)\right)^{b_l+1}.
\end{aligned}$$

We use Stirling's formula, $(b_l+1)! > \left(\frac{b_l+1}{e}\right)^{b_l+1}$, and $b_l! > \frac{(b_l+1)!}{2^{b_l+1}} > \left(\frac{b_l+1}{2e}\right)^{b_l+1}$. Also, we assume $2e\alpha < 1$.

$$\begin{aligned}
&\mathbf{E}[v(F)(1-\mu_{\vec{b}}(F))]\\
&\ge \frac{\alpha}{\lambda}\mathcal{V} - \frac{\beta}{\lambda}\mathcal{V}\sum_{l=1}^{d}\left(\frac{2e\alpha}{\lambda}\right)^{b_l+1} - \frac{\alpha}{\lambda}\mathcal{V}\sum_{l=1}^{d}\left(\frac{e\alpha}{\lambda}\right)^{b_l+1}\\
&\ge \frac{\alpha}{\lambda}\mathcal{V} - \frac{2e\alpha\beta}{\lambda}\mathcal{V}\sum_{l=1}^{d}\frac{1}{\lambda^{b_l+1}} - \frac{e\alpha^2}{\lambda}\mathcal{V}\sum_{l=1}^{d}\frac{1}{\lambda^{b_l+1}}\\
&= \frac{\alpha}{\lambda}\mathcal{V}(1-2e\beta-e\alpha).
\end{aligned}$$

We choose optimally $2e\alpha = -1+\sqrt{3}$ and $2e\beta = 2-\sqrt{3}$ which gives an approximation factor of $2e\lambda/(2-\sqrt{3}) < 21\lambda$. We can derandomize the algorithm using conditional expectations provided that all the b_l's are constant. In that case, we can evaluate $\mathbf{E}[v(F)(1-\mu_{\vec{b}}(F))]$ by summing a polynomial number of terms.

THEOREM 5.2. *For Stochastic b-matching with constant capacity $\vec{b}$, there is a polynomial-time algorithm which finds a set of items yielding expected value at least $ADAPT/21\lambda$ where $\lambda \ge 1$ is the solution of $\sum_j 1/\lambda^{b_j+1} = 1$. Therefore*

$$ADAPT \le 21\lambda\ NONADAPT.$$

This closes the adaptivity gap for b-matching up to a constant factor, since we know that it could be as large as $\lambda/4$. In case $\vec{b} = (B, B, \ldots, B)$, we get $\lambda = d^{1/(B+1)}$. This means that our approximation factors for Stochastic Set Packing and b-matching are near-optimal even in the deterministic case where we have hardness of $d^{1/(B+1)-\epsilon}$-approximation for any fixed $\epsilon > 0$.

6 Limitations when knowing only the mean sizes

Our algorithms only use knowledge of the truncated mean sizes of items. Here we show that this knowledge does not allow to determine the expected value of an optimal adaptive strategy to within a factor better than $\Theta(d)$ in the general case and $\Theta(\sqrt{d})$ in the Set Packing case.

Consider two instances of General Stochastic Packing with d items. Item i in the first instance has (deterministic) size $1/(2d)$ in all components $j \ne i$ and size 1 or 0 with probability $1/2$ in component i. In the second instance, item i has deterministic size equal to the expected size of item i in the first instance. In the second instance all d items fit, while in the first the expected number of items that we can fit is $O(1)$.

In the Set Packing case, consider two different instances with an infinite supply (or large supply) of the same item. In the first instance, an item has size $(1,1,\cdots,1)$ with probability $1/d$ and size $(0,0,\cdots,0)$ otherwise. In the second instance, an item has size e_i for $i = 1,\cdots,d$ with probability $1/d$. The expected size of an item is $(1/d, 1/d, \cdots, 1/d)$ in both instances. In the first instance, any policy will fit $2d-1$ items in expectation while in the second instance it will get $\Theta(\sqrt{d})$ items by the birthday paradox, for a ratio of $\Theta(\sqrt{d})$.

7 Restricted Stochastic Packing

As the last variant of Stochastic Packing, we consider instances where the item sizes are vectors restricted to $\vec{S}(i) \in [0,1]^d$ and the capacity vector is a given vector $\vec{b} \in \mathbb{R}^d_+$ with $b_j \geq 1$ for all j. Similarly to b-matching, we prove an approximation factor as a function of the capacity $\vec{b}$, and we find that our approach is particularly successful in case of capacity very large compared to item sizes.

THEOREM 7.1. *S For Restricted Stochastic Packing with item sizes $\vec{S}(i) \in [0,1]^d$ and capacity $\vec{b}$, there is a polynomial-time algorithm which finds a set of items yielding expected value at least $ADAPT/120\lambda$ where $\lambda \geq 1$ is the solution of $\sum_{i=1}^{d} 1/\lambda^{b_i} = 1$. I.e.,*

$$ADAPT \leq 120\lambda\ NONADAPT.$$

Proof. Consider the LP bounding the performance of any adaptive policy:

$$\mathcal{V} = \max \left\{ \sum_i x_i v_i : \begin{array}{ll} \sum_i x_i \mu_j(i) \leq b_j + 1 & \forall j \\ 0 \leq x_i \leq 1 & \forall i \end{array} \right\}.$$

Assume that $v_i \leq \frac{\beta}{\lambda}\mathcal{V}$ for each item i, for some constant $\beta > 0$ to be chosen later, otherwise one item alone is a good approximate solution. We find an optimal solution $\vec{x}$ and define

$$q_i = \frac{\alpha}{\lambda} x_i,$$

$\alpha > 0$ again to be chosen later.

Our randomized non-adaptive policy inserts item i with probability q_i. Let's estimate the probability that this random set of items F fits, with respect to both (independent) levels of randomization - our randomized policy and the random item sizes. For each j,

$$\mathbf{E}[S_j(F)] = \sum_i q_i \mu_j(i) \leq \frac{\alpha}{\lambda}(b_j + 1).$$

Since this is a sum of $[0,1]$ independent random variables, we apply the Chernoff bound (for random variables with support in $[0,1]$ as in [7], but we use the form given in Theorem 4.1 of [8] for the binomial case) to estimate the probability of overflow (use $\mu \leq \alpha(b_j + 1)/\lambda, 1 + \delta = b_j/\mu$):

$$\begin{aligned} \Pr[S_j(F) > b_j] &< \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\mu} < \left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu} \\ &\leq \left(\frac{e\mu}{b_j}\right)^{b_j} \leq \left(\frac{2e\alpha}{\lambda}\right)^{b_j} \end{aligned}$$

and using the union bound,

$$\Pr[\exists j; S_j(F) > b_j] < 2e\alpha \sum_{j=1}^{d} \frac{1}{\lambda^{b_j}} = 2e\alpha.$$

Now we estimate the probability that the value of F is too low. We assume that $v_i \leq \frac{\beta}{\lambda}\mathcal{V}$ and by scaling we obtain values $v'_i = \frac{\lambda}{\beta\mathcal{V}} v_i \in [0,1]$. We sample each of them with probability q_i which yields a random sum W with expectation $\mathbf{E}[W] = \sum_i q_i v'_i = \alpha/\beta$. Again by Chernoff bound (extension of Theorem 4.2 of [8]),

$$\Pr\left[W < \frac{1}{2}\mathbf{E}[W]\right] < e^{-\mathbf{E}[W]/8} = e^{-\alpha/8\beta}.$$

We choose $\alpha = 1/10$ and $\beta = 1/100$ which yields $\Pr[\exists j; S_j(F) > b_j] < 2e\alpha < 0.544$ and $\Pr[v(F) < \frac{1}{20\lambda}\mathcal{V}] < e^{-\alpha/8\beta} < 0.287$, which means that with probability at least 0.169, we get a feasible solution of value $\frac{1}{20\lambda}\mathcal{V}$. The expected value achieved by our randomized policy is at least $\frac{1}{120\lambda}\mathcal{V}$.

Finally, note that any randomized non-adaptive policy can be seen as a convex linear combination of deterministic non-adaptive policies. Therefore, there is also a deterministic non-adaptive policy achieving $NONADAPT \geq \frac{1}{120\lambda}\ ADAPT$. A fixed set F achieving expected value at least $ADAPT/120\lambda$ can be found using the method of pessimistic estimators applied to the Chernoff bounds, see [9].

8 Inapproximability of PIP

Here we improve the known results on the hardness of approximation for PIP. In the general case, it was only known [1] that a $d^{1/2-\epsilon}$-approximation, for any fixed $\epsilon > 0$, would imply $NP = ZPP$ (using a reduction from Max Clique). We improve this result to $d^{1-\epsilon}$.

THEOREM 8.1. *There is no polynomial-time $d^{1-\epsilon}$-approximation algorithm for PIP for any $\epsilon > 0$, unless $NP = ZPP$.*

Proof. We use Håstad's result on the inapproximability of Max Clique [6], or more conveniently maximum stable set. For a graph G, we define a PIP instance: Let $A \in \mathbb{R}^{d\times n}_+$ be a matrix where $d = n = |V(G)|$, $A_{ii} = 1$ for every i, $A_{ij} = 1/n$ for $\{i,j\} \in E(G)$ and $A_{ij} = 0$ otherwise. Let $\vec{b} = \vec{v} = (1, 1, \ldots 1)$. It is easy to see that $A\vec{x} \leq \vec{b}$ for $x \in \{0,1\}^n$ if and only if $\vec{x}$ is the characteristic vector of a stable set. Therefore approximating the optimum of this PIP to within $d^{1-\epsilon}$ for any $\epsilon > 0$ would imply an $n^{1-\epsilon}$-approximation algorithm for maximum stable set, which would imply $NP = ZPP$.

This proves that our greedy algorithm (Section 4) is essential optimal even in the deterministic case. Next we turn to the case of "small items", in particular $A \in [0,1]^{d\times n}$ and $\vec{b} = (B, B, \ldots, B)$, $B \geq 2$ integer. In this case, we have an $O(d^{1/B})$-approximation algorithm

(Section 7) and the known hardness result [1] was that a $d^{1/(B+1)-\epsilon}$-approximation would imply $NP = ZPP$. We strengthen this result to $d^{1/B-\epsilon}$.

THEOREM 8.2. *There is no polynomial-time $d^{1/B-\epsilon}$-approximation algorithm for PIP with $A \in [0,1]^{d\times n}$ and $\vec{b} = (B, B, \ldots, B), B \in \mathbb{Z}_+, B \geq 2$, unless $NP = ZPP$.*

Proof. For a given graph $G = (V, E)$, denote by d the number of B-cliques ($d < n^B$). Define a $d \times n$ matrix A (i.e., indexed by the B-cliques and vertices of G) where $A(Q, v) = 1$ if vertex v belongs to clique Q, $A(Q, v) = 1/n$ if vertex v is connected by an edge to clique Q, and $A(Q, v) = 0$ otherwise. Denote the optimum of this PIP with all values $v_i = 1$ by $\mathcal{V}$. Let $\epsilon > 0$ be arbitrarily small, and assume that we can approximate $\mathcal{V}$ to within a factor of $d^{1/B-\epsilon}$.

Suppose that $\vec{x}$ is the characteristic vector of a stable set $S, |S| = \alpha(G)$. Then $A\vec{x} \leq \vec{b}$ because in any clique, there is at most one member of S and the remaining vertices contribute at most $1/n$ each. Thus the optimum of the PIP is $\mathcal{V} \geq \sum_v x_v = \alpha(G)$.

If $A\vec{x} \leq \vec{b}$ for some $\vec{x} \in \{0,1\}^d$, then the subgraph induced by $S = \{v : x_v = 1\}$ cannot have a clique larger than B: suppose $R \subseteq S$ is a clique of size $B + 1$, and $Q \subset R$ is a sub-clique of size B. Then $(A\vec{x})(Q)$ (the component of $A\vec{x}$ indexed by Q) must exceed B, since it collects 1 from each vertex in Q plus at least $1/n$ from the remaining vertex in $R \setminus Q$. Finally, we invoke a lemma from [1] which states that a subgraph on $|S| = \sum_v x_v$ vertices without cliques larger than B must have a stable set of size $\alpha(G) \geq |S|^{1/B}$, i.e. $\mathcal{V} \leq (\alpha(G))^B$.

We assume that we can find $\mathcal{W}$ such that $\mathcal{V}/d^{1/B-\epsilon} \leq \mathcal{W} \leq \mathcal{V}$. Taking $a = \mathcal{W}^{1/B}$, we show that a must therefore be a $n^{1-\epsilon}$-approximation to $\alpha(G)$. We know that $a \leq \mathcal{V}^{1/B} \leq \alpha(G)$. On the other hand,

$$a \geq \left(\frac{\mathcal{V}}{d^{1/B-\epsilon}}\right)^{1/B} \geq \frac{\alpha(G)^{1/B}}{n^{1/B-\epsilon}} \geq \frac{\alpha(G)}{n^{1-\epsilon}}$$

where we have used $\mathcal{V} \geq \alpha(G)$, $d < n^B$, and finally $\alpha(G) \leq n$. This proves that a is an $n^{1-\epsilon}$-approximation to $\alpha(G)$, which would imply $NP = ZPP$.

9 PSPACE-hardness of Stochastic Packing

Consider the problem of finding the optimal adaptive policy. In [3], we showed how adaptive policies for Stochastic Knapsack are related to Arthur-Merlin games. This yields PSPACE-hardness results for certain questions; namely, whether it is possible to fill the knapsack exactly to its capacity with a certain probability, or what is the adaptive optimum for a Stochastic Knapsack instance with randomness in both size and value of each item. However, we were not able to prove that it is PSPACE-hard to find the adaptive optimum with deterministic item values.

In contrast to the inapproximability results of Section 8, here we do not regard dimension d as part of the input. Let us remark that for deterministic PIP, it is NP-hard to find the optimum but there is a PTAS for any fixed $d \geq 1$ [4]. For Stochastic Packing, this is impossible due to the adaptivity gap and limitation of knowing only the mean item sizes. However, consider a scenario where the probability distributions are discrete and completely known. Then we can consider finding the optimum adaptive policy *exactly*. Here we prove that this is PSPACE-hard even for Stochastic Packing with only two random size components and deterministic values.

For the PSPACE-hardness reduction, we refer to the following PSPACE-hard problem (see [2], Fact 4.1).

Problem: *MAX-PROB SSAT*

Input: Boolean 3-cnf formula $\Phi : \{0,1\}^{2k} \to \{0,1\}$ with variables $x_1, y_1, \ldots, x_k, y_k$.

$$P(\Phi) = \mathcal{M}x_1\mathcal{A}y_1\mathcal{M}x_2\mathcal{A}y_2 \ldots \mathcal{M}x_k\mathcal{A}y_k\Phi(x_1, y_1, \ldots x_k, y_k)$$

where $\mathcal{M}x\ f(x) = \max\{f(0), f(1)\}$
and $\mathcal{A}y\ g(y) = (g(0) + g(1))/2$.

Output: YES, if $P(\Phi) = 1$; NO, if $P(\Phi) \leq 1/2$.

THEOREM 9.1. *For Stochastic Packing in a fixed dimension $d \geq 2$, let $\hat{p}(V)$ be the maximum probability that an adaptive policy inserts successfully a set of items of total value at least V. Then for any fixed $\epsilon > 0$, it is PSPACE-hard to distinguish whether $\hat{p}(V) = 1$ or $\hat{p}(V) \leq 3/4$.*

Proof. We can assume that $d = 2$. We define a Stochastic Packing instance corresponding to a 3-cnf formula $\Phi(x_1, y_1, \ldots, x_k, y_k)$ of m clauses. The 2-dimensional sizes will have the same format in each component, $[VARS \mid CLAUSES]$, where $VARS$ have k digits and $CLAUSES$ have m digits. All digits are in base 10 to avoid any overflow. It will be convenient to consider the individual digits as "2-dimensional", with a pair of components indexed by 0 and 1. In addition, we define deterministic item values with the same format $[VARS \mid CLAUSES]$.

For each $i \in \{1, \ldots, d\}$, $x_i \in \{0,1\}$ and $f_i \in \{0,1\}$, we define a "variable item" $I_i(x_i, f_i)$ which has 4 possible random sizes indexed by two random bits $y_i, r_i \in \{0,1\}$:

$$\vec{s}(I_i(x_i, f_i), y_i, r_i) = [VARS(i, f_i, r_i)|CLAUSES(i, x_i, y_i)].$$

$VARS(i, f_i, r_i)$ have two (three for $i = 1$ and one for $i = k$) nonzero digits: the i-th most significant digit has a 1 in the f_i-component (in both components for $i = 1$, independently of f_1), and the $(i+1)$-th most significant digit has a 1 in the r_i-component, except for $i = k$. Note that the policy can *choose* in which component to place the f_i contribution (for $i > 1$), while the placement of the r_i contribution is *random*.

In $CLAUSES(i, x_i, y_i)$, we get a nonzero value in the digits corresponding to clauses in which x_i or y_i appears. Variable x_i contributes 1 to the digit in the 1-component if the respective clause is satisfied by the value of x_i, or in the 0-component if the clause is not satisfied. Similarly, y_i contributes to the clauses where it appears. If both x_i and y_i appear in the same clause, the contributions add up. The values of items $I_i(x_i, f_i)$ are defined as

$$val(I_i(x_i, f_i)) = [VAR(i) \mid 0]$$

where $VAR(i)$ contains a 1 in the i-th digit and zeros otherwise. Then we define fill-in items F_{ji} $(i = 0, 1)$ whose size only contains a 1 in the i-component for the j-th clause digit. For each j, we have 3 items of type F_{j0} and 2 items of type F_{j1}. Their values are

$$val(F_{ji}) = [0 \mid CLAUSE(j)]$$

which means a 1 marking the j-th clause. The capacity of the knapsack is

$$C = [11111111 \mid 33333333333333]$$

in each dimension and the target value is also

$$V = [11111111 \mid 33333333333333].$$

Assume that $P(\Phi) = 1$. We can then define an adaptive policy which inserts one item I_i for each $i = 1, 2, \ldots, k$ (in this order), choosing $f_{i+1} = 1 - r_i$ for each $i < k$. Based on the satisfying strategy for formula Φ, the policy satisfies each clause and then adds fill-in items to achieve value 1 in each digit of $VARS$ and value 3 in each digit of $CLAUSES$.

On the other hand, assume $P(\Phi) \leq 1/2$. Any adaptive policy inserting exactly 1 item I_i for each i and abiding by the standard ordering of items can achieve the target value only if all clauses are properly satisfied (because otherwise it would need 3 items of type F_{j1} for some clause), and that happens with probability at most 1/2. However, we have to be careful about "cheating policies". Here, "cheating" means either inserting I_i after I_{i+1} or not inserting exactly 1 copy of each I_i. Consider a cheating policy and the first i for which this happens. In case I_i is not inserted at all, the policy cannot achieve the target value for $VARS$. In case more than 1 copy of I_i is inserted or I_i is inserted after I_{i+1}, there is 1/2 probability of overflow in the $VARS$ block of capacity. This is because the contribution of I_i to the $(i+1)$-th digit of $VARS$ hits a random component, while one of the two components would have been filled by I_{i+1} or another copy of I_i already. Either way, this leads to a failure with probability at least 1/2, conditioned on the event of cheating. In the worst case, the probability of success of a cheating policy can be 3/4.

THEOREM 9.2. *For a 2-dimensinal stochastic knapsack instance, it is PSPACE-hard to maximize the expected value achieved by an adaptive policy.*

Proof. We use the reduction from the previous proof. The maximum value that any policy can achieve is $V = [11111111 \mid 33333333333333]$. In case of a YES instance, an optimal policy achieves V with probability 1, whereas in case of a NO instance, it can succeed with probability at most 3/4. Therefore the expected value obtained in this case is at most $V - 1/4$.

References

[1] C. Chekuri and S. Khanna: On multidimensional packing problems, SIAM J. Computing 33:837–851, 2004.

[2] A. Condon, J. Feigenbaum, C. Lund and P. Shor: Random debaters and the hardness of approximating stochastic functions, SIAM J. Comp. 26:369–400, 1997.

[3] B. Dean, M. X. Goemans and J. Vondrák: Approximating the stochastic knapsack: the benefit of adaptivity. To appear in FOCS, 2004.

[4] A.M. Frieze and M.R.B. Clarke: Approximation algorithms for the m-dimensional 0-1 knapsack problem: worst-case and probabilistic analyses. European J. Oper. Res. 15:100–109, 1984.

[5] M. Halldórsson: Approximations of weighted independent set and hereditary subset problems. J. Graph Algorithms and Applications 4 (1):1–16, 2000.

[6] J.Håstad: Clique is hard to approximate to within $n^{1-\epsilon}$. In FOCS:627–636, 1996.

[7] W. Hoeffding: Probability inequalities for sums of bounded random variables. Amer. Stat. Assoc. J. 58:13–30, 1963.

[8] R. Motwani and P. Raghavan: Randomized Algorithms, Cambridge University Press 1995.

[9] P. Raghavan: Probabilistic construction of deterministic algorithms: approximating packing integer programs. J. Comp. and System Sci. 37:130–143, 1988.

[10] P. Raghavan and C. D. Thompson: Randomized rounding: a technique for provably good algorithms and algorithmic proofs. Combinatorica 7:365–374, 1987.

[11] A. Srinivasan. Improved approximations of packing and covering problems. In STOC:268–276, 1995.