# Augmenting Packing Dynamic Programs to Handle (Many) Additional Budget Constraints

Alexander Armbruster[*] Fabrizio Grandoni[†] Antoine Tinguely[†] Andreas Wiese[*]

**Abstract.** In a packing problem, we are given a collection $I$ of $n$ items, each one with a given profit. Our goal is to compute a maximum profit subset of these items that satisfies a given set of packing constraints which depend on the problem at hand. Several approximation algorithms for well-studied NP-hard packing problems are based on a reduction to an auxiliary (packing) problem which is then solved with a dynamic program (DP). Examples for this include approximation algorithms for Knapsack, Geometric Knapsack, Independent Set of Rectangles, and Maximum Throughput Scheduling.

Often, it is desirable to impose additional *global* constraints to packing problems, e.g., due to fairness considerations, diversification, quotas, or limited resources. More specifically, suppose that we are given $M$ additional *cardinality* or *budget constraints*, i.e., each constraint $h$ imposes for any solution $S \subseteq I$ the condition $\sum_{i\in S} a_{i,h} \le b_h$ for a given budget $b_h$ and values $a_{i,h}$ (which are binary for cardinality constraints). In this paper, we address the question how to efficiently handle such constraints. There is a naive way to modify a DP for a packing problem so that it incorporates them. However, this may increase the running time of the DP by a factor $\Omega(n)$ *per constraint*; hence, this yields a polynomial running time bound only for $M = O(1)$.

Our first result is a *meta-algorithm* that, given an exact DP for a packing problem, derives a (randomized) PTAS, i.e., a $(1+\varepsilon)$-approximation for any constant $\varepsilon > 0$, for the same packing problem with additionally $M = O(\log n/\log\log n)$ cardinality constraints. Our running time is of the form $f(M,\varepsilon)\cdot n^{O(1)}$, i.e., we obtain an *efficient parameterized approximation scheme* for the parameter $M$. Then, we extend our result to $M = O(\log n/\log\log n)$ budget constraints if all values $a_{i,h}$ are sufficiently small compared to the corresponding budget $b_h$ or if we are allowed to use resource augmentation. As an application, we transform several approximation algorithms for packing problems from the literature to approximation algorithms for the same respective problem augmented with $O(\log n/\log\log n)$ cardinality or budget constraints, while losing only a factor of $1+\varepsilon$ in the approximation ratio.

We complement our meta-algorithms with almost matching hardness results showing that, under the Gap Exponential-Time Hypothesis (Gap-ETH), a polynomial-time $(1+\varepsilon)$-approximation is not possible for $M = \omega(\log n)$ cardinality constraints or for $M = \omega(1)$ budget constraints without resource augmentation.

## 1 Introduction

Packing problems are an important and well-studied class of problems in combinatorial optimization. In these problems we are given a set of $n$ items $I$, where each item $i \in I$ has an associated profit $p(i) \in \mathbb{N}$. Our goal is to find a *feasible solution* $\mathsf{OPT} \subseteq I$ of maximum profit $\mathsf{opt} := p(\mathsf{OPT})$, where for any set of items $I' \subseteq I$ we define $p(I') := \sum_{i\in I'} p(i)$. The set of feasible solutions, denoted by $\mathcal{F}$, depends on the specific packing problem and on the concrete given instance. We assume that $\mathcal{F}$ is *downward closed*, i.e., if $R \subseteq S$ and $S \in \mathcal{F}$, then also $R \in \mathcal{F}$. For example, in the famous Knapsack problem each item $i$ has a size $s(i) \ge 0$, and a set $S \subseteq I$ is feasible if and only if $\sum_{i\in S} s(i) \le C$ for a given value $C$.

Many packing problems can be solved (exactly or approximately) via a type of dynamic program (DP) that we denote as a *packing DP* in this paper. Intuitively, in each state of a packing DP we make a choice and, depending on our choice, we select a set of items and continue in one or more different states (a formal definition is given in Section 2). A simple illustrative example is a straight-forward pseudo-polynomial time DP for the Knapsack

[*]Technical University of Munich, Germany (alexander.armbruster@tum.de, andreas.wiese@tum.de). Alexander Armbruster and Andreas Wiese were funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 551896423.

[†]IDSIA, USI-SUPSI, Switzerland (fabrizio.grandoni@supsi.ch, antoine.tinguely@supsi.ch). Fabrizio Grandoni was partially supported by the Swiss National Science Foundation (SNSF) Grants 200021-200731/1 and 200021-236706. Antoine Tinguely was supported by the SNSF Grant 200021-200731/1.

problem. Each state is characterized by an item $i$ and a residual knapsack capacity $C'$. The subproblem of such a state $(i, C')$ is to compute a subset of the items $\{1, \ldots, i\}$ of maximum total profit whose total size is upper bounded by $C'$. Given the state $(i, C')$ our choices are (a) to select $i$ and continue in the state $(i-1, C'-s(i))$ (assuming that $s(i) \leq C'$) and (b) not to select $i$ and continue in the state $(i-1, C')$. We can easily compute an optimal solution for each state. Then, our solution for the *root* state $(n, C)$ is an optimal solution to the given instance.

A common approach to design approximation algorithms for NP-hard packing problems is to first reduce the given problem to an auxiliary packing problem (maybe losing a small factor in the approximation ratio thereby) and then to invoke a packing DP for the latter problem. For example, by losing a factor $1+\varepsilon$ in the approximation, one can reduce a Knapsack instance to another instance in which the item sizes are polynomially bounded integers, and then solve the latter instance exactly with the above DP. Many algorithms for packing problems can be analyzed in this framework, e.g., for Geometric Knapsack [29, 22, 23, 37], Independent Set of Rectangles [2, 24], Independent Set of Polygons [26, 51], or Maximum Throughput Scheduling [4] and variations and generalizations of those problems.

In practice, it is often desirable to satisfy additional *global* constraints which capture, e.g., fairness considerations [20, 48], diversification [8] or quotas [1, 21]. More specifically, suppose we are given a generic packing problem Pack that we can solve exactly via a packing DP. In addition, assume that we are given $M$ *cardinality constraints*. For each $h \in \{1, \ldots, M\}$, the $h$-th cardinality constraint is specified by a subset of items $I_h \subseteq I$ and a budget $b_h \in \{1, \ldots, |I_h|\}$. The constraint imposes that each feasible solution $S$ may select at most $b_h$ items from $I_h$, i.e., $|S \cap I_h| \leq b_h$. Notice that the sets $I_h$ might overlap. We denote by $\text{Pack}_C^+$ the variant of Pack with the additional cardinality constraints. More generally, we might impose instead $M$ additional *budget constraints* where for each $h \in \{1, \ldots, M\}$, the $h$-th budget constraint is specified by a budget $b_h \in \mathbb{N}$ and values $a_{i,h} \in \mathbb{N}_0$ for each item $i \in I$. This constraint $h$ imposes the inequality $\sum_{i \in S} a_{i,h} \leq b_h$ for each feasible solution $S$. Note that cardinality constraints can be cast as budget constraints where $a_{i,h} \in \{0, 1\}$ for each item $i \in I$ and each constraint $h$. We denote by $\text{Pack}_B^+$ the resulting problem. When the type of constraint is clear from the context, we will simply use the term $\text{Pack}^+$. In both cases, let $\mathsf{OPT}^+$ be an optimal solution to the obtained problem and we define $\mathsf{opt}^+ := p(\mathsf{OPT}^+)$. In this paper, we address the following question:

*Given an exact packing DP for Pack, for which values of $M$ can one derive a polynomial-time algorithm that approximates $\text{Pack}^+$ within a factor $1+\varepsilon$ for any constant $\varepsilon > 0$?*

**1.1 Our Results** Our first result is that we answer the above question affirmatively for up to $M = O(\frac{\log n}{\log \log n})$ cardinality constraints. In other words, we present a *meta-algorithm* that, given as input an exact polynomial time packing DP for Pack, produces a PTAS for $\text{Pack}_C^+$ for these values of $M$. More precisely, assuming that $\tilde{n}$ is the number of bits in the input, we obtain the following result.

Theorem 1.1. *Given an exact packing DP for Pack with running time $T(\tilde{n})$ and a value $\varepsilon \in (0, 1/12)$, there is a randomized $(1+\varepsilon)$-approximation algorithm for $\text{Pack}_C^+$ whose running time is bounded by $(\log(Mn)/\varepsilon)^{O(M)}(nT(\tilde{n}))^{O(1)}$.*

By standard arguments, our running time is upper-bounded by $f(M, \varepsilon)(nT(\tilde{n}))^{O(1)}$ for a computable function $f$.[1] Therefore, if $T(\tilde{n}) \leq \tilde{n}^{O(1)}$, we obtain an *efficient parameterized approximation scheme* for $\text{Pack}^+$ with parameter $M$.

We obtain even better results for the important special class of *unary* DPs (see Section 3 for a formal definition). Intuitively, these are DPs in which for each state, the corresponding value depends only on the (already computed) value of at most one other state for each possible choice. Many DPs are of this type, e.g., the DP for Knapsack mentioned before. We obtain a further improvement for *disjoint* cardinality constraints, i.e., if $I_a \cap I_b = \emptyset$ for every distinct $a, b \in \{1, \ldots, M\}$. This arises naturally when one imposes quotas on subsets of items that form a partition, e.g., in gender quotas.

Theorem 1.2. *Given an exact unary packing DP for Pack with running time $T(\tilde{n})$ and a value $\varepsilon \in (0, 1/12)$, there is a randomized $(1+\varepsilon)$-approximation algorithm for $\text{Pack}_C^+$ with running time $(\log M/\varepsilon)^{O(M)}(nT(\tilde{n}))^{O(1)}$. For disjoint cardinality constraints, the running time is bounded by $(1/\varepsilon)^{O(M)}(nT(\tilde{n}))^{O(1)}$.*

[1] We have $(\log n)^k = o(n)$ for any fixed $k \in \mathbb{N}$. Using this, one can easily show that there is a computable function $g$ such that $(\log n)^k \leq g(k) \cdot n$ for any $k \in \mathbb{N}$ and $n \in \mathbb{N}$.

Therefore, assuming again $T(\tilde{n}) \leq \tilde{n}^{O(1)}$, in the unary case our algorithm runs in polynomial time if $M = O(\frac{\log n}{\log\log\log n})$; if the constraints are disjoint, this holds even for $M = O(\log n)$.

One might wonder whether our approach can be improved to handle much larger values of $M$. We show that this is impossible, even in very simple special cases, assuming the *(Randomized) Gap Exponential-Time Hypothesis (Gap-ETH)* [10, 14, 42].

THEOREM 1.3. *There exists a packing problem* PACK *that can be solved exactly via a polynomial-time unary packing DP such that, for every function $f(n) = \omega(\log n)$, the following holds. There exists a set of $f(n)$ disjoint cardinality constraints, all with budget 1, such that the resulting problem* $\text{PACK}_C^+$ *does not admit a (randomized) PTAS, assuming the (Randomized) Gap-ETH.*

Hence, the second result in Theorem 1.2 is (asymptotically) tight, and the result in Theorem 1.1 and the first result in Theorem 1.2 are almost tight. If instead of cardinality constraints we are given $M$ budget constraints, we prove that we *cannot* obtain a (randomized) PTAS, already when $M = \omega(1)$, again assuming the (Randomized) Gap-ETH. This is based on a reduction from $M$-dimensional Vector Knapsack and [15].

THEOREM 1.4. *There exists a packing problem* PACK *that can be solved exactly via a polynomial-time unary packing DP such that adding $\omega(1)$ budget constraints results in a problem* $\text{PACK}_B^+$ *that does not admit a (randomized) PTAS, even with unit profits, assuming the (Randomized) Gap-ETH.*

However, we prove that we *can* handle up to $M = O(\frac{\log n}{\log\log n})$ budget constraints if, for each constraint $h$, each coefficient $a_{i,h}$ is sufficiently small compared to the budget $b_h$, or if we allow $(1+\varepsilon)$-resource augmentation, i.e., we allow a slightly larger budget of $(1+\varepsilon)b_h$ for each constraint $h$ (but compare our objective function value with an optimal solution for the given budget constraints).

THEOREM 1.5. *Given an exact packing DP for* PACK *with running time $T(\tilde{n})$ and a value $\varepsilon \in (0, 1/12)$, there is a randomized $(1+\varepsilon)$-approximation algorithm for* $\text{PACK}_B^+$ *assuming either that*

*(1)* $a_{i,h} \leq O\big(\frac{\varepsilon^3}{\log M}\big)b_h$ *for each budget constraint $h$ and for each item $i \in I$, or*

*(2) under $(1 + O(\varepsilon))$-resource augmentation.*

*Its running time satisfies the same bounds as in Theorem 1.2 if the DP is unary, and the same as in Theorem 1.1 otherwise.*

## 1.2 Our Techniques

The heart of our approach in all our algorithms (due to Theorems 1.1, 1.2, and 1.5) is, at a very high level, a *randomized sketching* technique. In a nutshell, there is a naive way to add cardinality or budget constraints to the given packing DP for PACK by augmenting the states with *counters* that keep track of the total remaining budget of each constraint $h$. However, this increases the running time by $\Theta(\text{poly}(b_h))$ for each constraint $h$, which might be $\Omega(\text{poly}(n))$ *per constraint.* Hence, this can work only for $M = O(1)$. Instead, we exploit more compact *randomized counters* that intuitively keep the measured quantities close to their expected values with only small variance. As a result, we increase our running time only by a factor of $(\log(Mn)/\varepsilon)^{O(1)}$ per constraint which allows us to handle $M = O(\log n/\log\log n)$ constraints in polynomial time (and even a bit more for unary DPs). On the other hand, we argue that the resulting variance is sufficiently small such that we lose only a factor of $1+\varepsilon$ in our approximation ratio due to this.

Let us first illustrate in more detail how our approach works for the packing DP for Knapsack mentioned above, starting with the naive method. For simplicity, let us consider the case of a single cardinality constraint defined via one set $I_1$ and one budget $b_1$, i.e., $M = 1$. The idea is to expand each state $(i, C')$ by adding a value $b \in \{0, \ldots, b_1\}$ which indicates how many items from $I_1$ one is allowed to select. Hence, the subproblem of the resulting state $(i, C', b)$ is to compute a subset of the items $\{1, \ldots, i\}$ *containing at most $b$ items from $I_1$* whose total size is upper bounded by $C'$. Again, the objective is to maximize the total profit of these items. In the state $(i, C', b)$, we still have the two options to select and not to select $i$. However, now we are not allowed to select $i$ if $i \in I_1$ and $b = 0$; in that case, our only option is not to select $i$ and, thus, continue with the state $(i-1, C', b)$. If we select $i$ and $i \in I_1$, then the next state is $(i-1, C'-s(i), b-1)$ since we can select at most $b-1$ items from $I_1$ in the following. If we select $i$ and $i \notin I_1$ then we simply continue with the state $(i-1, C'-s(i), b)$. Then, the computed solution in the root state $(n, C, b_1)$ has a value of $\mathsf{opt}^+$. It is easy to extend this idea to a generic value of $M$. However, the number of states (and hence the running time) might grow by a factor $b_h = \Theta(n)$ per constraint, and hence $\Theta(n^M)$ in total, which is polynomial only for $M = O(1)$.

Therefore, we use a different approach which we illustrate now for the setting of Knapsack and $M = 1$. Let $\delta := (\varepsilon/\log(Mn))^{O(1)}$, so that $1/\delta$ is integer. Assume that $b_1 \geq 1/\delta$ as otherwise the trivial approach above is sufficient. Instead of updating $b$ each time one item from $I_1$ is selected, we keep an approximate randomized counter of how many items from $I_1$ we may still select. In more detail, when in a given state $(i, C', b)$ of the DP one of the choices is to select an item $i \in I_1$, instead of decreasing the budget $b$ by 1 *deterministically*, we decrease it by $\delta b_1$ with probability $\frac{1}{\delta b_1}$, and by 0 otherwise (so that *in expectation* the decrease is 1). Our DP still computes one (deterministic) choice for $(i, C', b)$ but this choice may lead randomly to one of two states, each with a certain probability. The value associated to this choice is the expected value obtained by the above process, i.e., the profit $p(i)$ of $i$ plus $\frac{1}{\delta b_1}$ times the value of $(i-1, C'-s(i), b-\delta b_1)$ plus $1 - \frac{1}{\delta b_1}$ times the value of $(i-1, C'-s(i), b)$. Altogether, we obtain a DP where the computed value of each state is the expected profit according to the above random process, assuming that in each state we select (deterministically) the choice that maximizes the above expectation.

With concentration arguments, we can show that the resulting expected profit when starting in the root state $(n, C, b_1)$ is at least $(1-\varepsilon)\mathsf{opt}^+$. Thus, even though we introduced randomness and, hence, variance in the process, this did not decrease our expected profit by much. On the other hand, all possible values of $b$ we can reach this way are in $\mathcal{B} := \{0, \delta b_1, 2\delta b_1, \ldots, b_1\}$. Thus, we can restrict ourselves to states of the form $(i, C', b)$ where $b \in \mathcal{B}$. This increases our number of states only by a factor of $|\mathcal{B}| = (\log(Mn)/\varepsilon)^{O(1)}$ instead of $\Theta(n)$.

Contrary to the naive implementation, this does not naturally yield an unique (optimal) *solution* for the root state $(n, C, b_1)$ since the state transition is randomized. However, we can compute *some* solution by starting in the root state and performing the random process described above, such that in each state, we make the (deterministic) optimal choice computed by the DP. In expectation, this solution will have a profit of at least $(1-\varepsilon)\mathsf{opt}^+$. Also, with concentration arguments we can show that w.h.p. the output contains at most $(1+\varepsilon)b_1$ items from $I_1$, i.e., it is "almost" feasible. Hence we can make it feasible w.h.p. by dropping each item independently with probability $2\varepsilon$. Altogether, we can show that we obtain a feasible solution with an expected profit of at least $(1-O(\varepsilon))\mathsf{opt}^+$.

*Extension to non-unary DPs.* The above approach generalizes naturally to larger values of $M$ and to other unary packing DPs (indeed, it even works with a larger value of $\delta$). However, there are also packing DPs where the value of a (non-base) state $s$ for a given choice $c$ depends on the values of two or more subsequent states. Let us assume for simplicity that there are exactly two such states $s_{c,1}$ and $s_{c,2}$, and that the choice $c$ corresponds to the selection of a subset of items $I(c)$ with $|I(c)| \leq 1$ (we will prove later that we can assume this w.l.o.g.). Again, it is instructive to consider first the naive approach for $M = 1$. Similarly as before, for each state $s$ of the given DP we create a state $(s, b)$ for each value $b$ of the remaining budget for selecting further items from $I_1$. However, after making a choice $c$, now the DP needs also to select how the remaining budget of $b - |I(c) \cap I_1|$ for items from $I_1$ is distributed on the two subsequent states $s_{c,1}$ and $s_{c,2}$. This leads to two states $(s_{c,1}, b_{c,1})$ and $(s_{c,1}, b_{c,2})$ on which we recurse independently. Notice that the optimal choices induce a binary tree $T$ rooted in the root state of the DP and in which the base states form the leaves.

The approach above increases the number of states and also the number of choices per state by a factor of $\Theta(n)$ which yields polynomial running time only when $M = O(1)$. To allow also larger values of $M$, our approach from the unary case does not work unfortunately: the optimal solution may repeatedly split the remaining budget of some constraint into two equal halves such that it reaches $1/\delta$ states with a budget of $\delta b_1$ each. If we select an item from $I_1$ in such a state, then for the residual budget there are only two possible values: $\delta b_1$ and 0. In particular, the variance is very large and we can no longer guarantee that (in expectation) we can select almost $\delta b_1$ items from $I_1$ in this state and its subsequent states before the residual budget drops to 0. Therefore, we allow more values for the residual budgets for each constraint. Roughly speaking, our allowed budgets are grouped into intervals of the form $(2^\ell, 2^{\ell+1}]$ with $\ell \in \mathbb{N}$, and in each such interval we allow only budgets which are multiples of $\delta 2^\ell$. This yields $O(\log n/\delta)$ different budget values which is small enough. A crucial observation is that the states with budgets in the same group $(2^\ell, 2^{\ell+1}]$ form paths in the tree $T$. Therefore, intuitively, on those paths we can use a similar argumentation as in the unary case. However, we have to handle the case that in $T$ there is a state $(s, b)$ with $b \in (2^\ell, 2^{\ell+1}]$ and, ideally, we would like to recurse on a child state $(s_{c,j}, b_{c,j})$ with $b_{c,j} \leq 2^\ell$. Then, $b_{c,j}$ must be in a smaller group than $b$ and it might be that $b_{c,j}$ is not among the set of allowed budgets of that group. Therefore, we round it down to the largest allowed budget $b'_{c,j} \leq b_{c,j}$. This introduces an error. However, our budgets are sufficiently dense such that we lose at most a factor of $1+\delta$ in our remaining budget by using $b'_{c,j}$ instead of $b_{c,j}$, i.e., when going from $(s, b)$ to $(s_{c,j}, b'_{c,j})$. This can happen at most $O(\log b_1) \leq O(\log n)$ many times on a path from the root of $T$ to a leaf; hence, the total loss is bounded by $(1+\delta)^{O(\log n)} \leq 1+O(\varepsilon)$.

We remark that a natural alternative approach would be to allow only powers of $1+\delta$ as residual budgets and rounding down each arising remaining budget $b_{c,j}$ deterministically to the next smaller power $b'_{c,j}$ of $1+\delta$ (similar to [6, 33, 39]). However, this does not work since $T$ might contain root-to-leaf paths whose length is linear in $n$. Then, the total loss would be $(1+\delta)^{\Omega(n)}$ which is too large.

*Budget Constraints.* We extend the approach described above to budget constraints. Previously, when we selected an item $i$ with $i \in I_h$, we would have liked to decrease the budget $b_h$ by 1 deterministically, but instead we decreased it by $\delta b_h$ with probability $\frac{1}{\delta b_h}$. For a budget constraint $h$ we would like to decrease $b_h$ deterministically by $a_{i,h}$ when we select $i$. Instead, we decrease now $b_h$ by $\delta b_h$ with the same probability as before $a_{i,h}$ *times independently.*[2] With similar arguments as in the cardinality case we prove that with probability at least $1-O(\varepsilon)$ our computed solution $\mathsf{APX}^+$ has a profit of at least $(1-\varepsilon)\mathsf{opt}^+$ and satisfies each budget constraint $h$ under resource augmentation, i.e., $\sum_{i\in\mathsf{APX}^+} a_{i,h} \leq (1+\varepsilon)b_h$. Unlike the cardinality case, it is no longer sufficient to drop each item with probability $O(\varepsilon)$ to obtain a feasible solution with probability close to one. This is due to the presence of large coefficients $a_{i,h}$ w.r.t. the respective budget $b_h$, e.g., it might happen that $a_{i,h} = b_h$ for some item $i \in \mathsf{APX}^+$. However, this is not a limitation of our approach only. Indeed, in Theorem 1.4 we prove for budget constraints that resource augmentation is necessary to handle any (nontrivial) number $M = \omega(1)$ of constraints.

On the other hand, if each coefficient $a_{i,h}$ is sufficiently small compared to the respective budget $b_h$, then we show that resource augmentation is *not* necessary. In more detail, we prove that we can reduce the setting *without* resource augmentation to the setting *with* resource augmentation. Intuitively, we drop each item from $\mathsf{OPT}$ independently with probability $O(\varepsilon)$; using concentration arguments, we show that the remaining items still have a total profit of $(1-\varepsilon)\mathsf{opt}^+$ and satisfy each budget constraint $h$ even for a reduced budget bound of $(1-\varepsilon)b_h$ (with some positive probability). Then, it is sufficient to apply an algorithm with resource augmentation using the reduced budgets, hence obtaining a feasible solution w.r.t. the original budgets.

**1.3 Applications** Applying our framework to exact algorithms is straightforward; one needs to verify that the initial problem is a packing problem (in particular, that the feasibility set of every instance is downward closed) and make sure that the DP at hand matches our model as described in Section 2 (or adapt it accordingly).

In many cases, our framework can also be applied if the given packing DP is an approximation algorithm, that is, we preserve the DP's approximation ratio (up to a factor $1+\varepsilon$) while adding up to $O\big(\frac{\log n}{\log\log n}\big)$ cardinality constraints. This can be done as follows. Approximation algorithms for many packing problems use a packing DP to compute solutions that obey certain structural properties, e.g., the items are chosen and placed in a specific pattern. More precisely, typically this DP computes the *optimal* solution with these structural properties. Then, one argues that this yields an $\alpha$-approximation by showing that, given the optimal solution $\mathsf{OPT}$ of the given instance, one can construct an $\alpha$-approximate solution satisfying these properties such that the items of this solution are a subset of $\mathsf{OPT}$. Then, our meta-algorithm can be applied which yields a DP that computes an $(1+\varepsilon)$-approximation for the problem to compute a solution that satisfies the mentioned structural properties and the additional cardinality or budget constraints. We can argue that this yields a $\alpha(1+\varepsilon)$-approximation algorithm for the initial problem augmented by these additional constraints: consider an auxiliary instance whose input consists only of the optimal solution $\mathsf{OPT}^+$ to the latter problem. This is in particular a solution to the packing problem *without* the additional constraints. Hence, for this instance, there is a $\alpha$-approximate solution $\mathsf{APX}^+ \subseteq \mathsf{OPT}^+$ that satisfies the mentioned structural properties, i.e., $p(\mathsf{APX}^+) \geq p(\mathsf{OPT}^+)/\alpha$. Since $\mathsf{APX}^+ \subseteq \mathsf{OPT}^+$, also $\mathsf{APX}^+$ satisfies the additional constraints. Therefore, also for our new problem *with* the additional constraints, there exists the $\alpha$-approximate solution $\mathsf{APX}^+$ satisfying the mentioned structural properties. Therefore, our augmented DP computes a solution with an approximation ratio of $\alpha(1+\varepsilon)$.

With this reasoning, we obtain approximation algorithms with additional cardinality or budget constraints for the problems listed in Table 1.1. For all the listed problems, we match the approximation ratio of the best known algorithm without additional constraints up to a factor of $1+\varepsilon$. For example, our results yield a PTAS for Knapsack with $O(\log n/\log\log\log n)$ additional cardinality constraints. The details of this application and the needed adaptations for the other problems listed in Table 1.1 can be found in the full version of this paper. To the best of our knowledge, there was no PTAS known for Knapsack with $\omega(1)$ (unstructured) cardinality constraints, despite a lot of research on generalizations of Knapsack [9, 41, 44, 12, 16, 17, 18, 19, 28]. Also, our technique may

[2]Thanks to a preprocessing step, we ensure that all coefficients $a_{i,h}$ are polynomially bounded integers.

Table 1.1: Overview of approximation algorithms achieved by applying our framework.

| **Approx. ratio** | **Problem** | **Packing DPs** |
|---|---|---|
| $1+\varepsilon$ | Knapsack | [30, 50] |
| $2+\varepsilon$ | Maximum Independent Set of Rectangles | [25, 45] |
| $\frac{8d}{3}+\varepsilon$ | Maximum Independent Set of $d$-oriented Polygons | [26] |
| $1+\varepsilon$ | Maximum Weight Independent Set of $\nu$-Shrinkable Convex Polygons | [51] |
| $1+\varepsilon$ | Guillotine Separable Rectangles | [2, 38] |
| $\frac{17}{9}+\varepsilon$ | Weighted 2D-Knapsack | [22] |
| $1+\varepsilon$ | Maximum Throughput Scheduling with Fixed Start and End Times | [4] |

be useful for future work in order to add global cardinality or budget constraints to a packing DP in a black-box manner.

**1.4 Other related work** Our technique for incorporating budget constraints into DPs has some similarities with the approach used in [27] to derive a PTAS for the Unsplittable Flow on a Path problem (UFP). In UFP one is given a path with edge capacities and a collection of tasks, each one characterized by a subpath, a demand and a profit. The goal is to compute a maximum profit subset of tasks such that the total demand of the selected tasks on each edge is within the respective capacity. Intuitively, the authors design a packing DP in which the demand of selected tasks of relatively small demand is rounded up or down randomly, similarly to what we do. However, each selected task uses a very simple structured set of capacity constraints which are those corresponding to its subpath. It is unclear how to extend the approach in [27] to generic packing DPs with additional unstructured cardinality or budget constraints. In particular, in our algorithm we need to split the budget of some constraint possibly many times over up to recursion $\Omega(n)$ levels, yielding new and non-trivial constraints for each subproblem. This is one of our main technical contributions and it is entirely new.

Adding cardinality or budget constraints to packing problems is a well-studied area. There are several PTASes or even FPTASes for, e.g., Budgeted Matching [7], Multiple Knapsack [11, 34, 35], Multiple Choice Knapsack (Knapsack with disjoint cardinality constraints) [5, 40, 49], and others [36]. There are results in which DP formulations of problems are transformed to obtain other results for these problems. For example, in [52], Woeginger obtains FPTASs for scheduling problems this way. Phrus and Woeginger [47] describe a generic method to obtain FPTASs for packing problems admitting an exact DP with a restricted type of running time: the latter condition is not satisfied by the packing problems with $M=\omega(1)$ additional cardinality constraints considered here. In addition, some (packing) DPs can be reformulated as integral linear programs [43], which can be leveraged to formulate linear programming relaxations with improved integrality gaps [3]. The idea of sparsifying the number of values to keep track of in a DP has also been used for parametrized approximations for graph problems [6, 33]. Randomized counters, similar to the ones that we use, are also extensively used in the streaming literature, e.g., for frequency estimation (see, e.g., the survey [13]). In that case, however, the main goal is to save space.

## 2 Formalization of Packing Dynamic Programs

We now formalize a given packing DP for an arbitrary packing problem PACK. Throughout this paper, for any $k\in\mathbb{N}$, we define $[k]:=\{1,\dots,k\}$ and we denote by log the logarithm of base 2.

We assume that there is a polynomial-time algorithm $A_{\text{PACK}}$ that, given an instance of PACK with input items $I=\{1,...,n\}$, computes a polynomial-size set of *states* $\mathcal{S}$, a partial order $\succ$ for the states and, for each state $s\in\mathcal{S}$, a polynomial-size set of *choices* $CH(s)$. There is a special set of *base states* $\mathcal{S}_{\text{base}}\subseteq\mathcal{S}$ such that $CH(s)=\emptyset$ for $s\in\mathcal{S}_{\text{base}}$. There is also a special *root* state $r\in\mathcal{S}$. We assume that the states and choices defined by $A_{\text{PACK}}$ are oblivious to the profits of the items in $I$. Each choice $c\in CH(s)$ for a state $s\in\mathcal{S}\setminus\mathcal{S}_{\text{base}}$ is characterized by a (possibly empty) subset of items $I(c)\subseteq I$ (which we, intuitively, add to the solution if we make the choice $c$) and a sequence of children states $s_{c,1},\dots,s_{c,d(c)}$ for some $d(c)\in\mathbb{N}$ such that each state $s_{c,j}$ with $j\in[d(c)]$ satisfies $s\succ s_{c,j}$.

Given the states and choices as defined above, in polynomial time $A_{\text{PACK}}$ computes the following quantities $\mathsf{OPT}(s)$ and $\mathsf{opt}(s)=p(\mathsf{OPT}(s))$ for each $s\in\mathcal{S}$, following the partial order induced by $\succ$. For each $s\in\mathcal{S}_{\text{base}}$, it sets $\mathsf{opt}(s) = 0$ and $\mathsf{OPT}(s) = \emptyset$. For each non-base state $s$, it sets

$$\mathsf{opt}(s) := \max_{c \in CH(s)} \{p(I(c)) + \mathsf{opt}(s_{c,1}) + \ldots + \mathsf{opt}(s_{c,d(c)})\}. \tag{2.1}$$

Let $c^* \in CH(s)$ be the choice that achieves the maximum above; then $A_{\text{PACK}}$ sets $\mathsf{OPT}(s) := I(c^*(s)) \cup \bigcup_{j=1}^{d(c^*)} \mathsf{OPT}(s_{c^*,j})$ using the previously computed values $\mathsf{OPT}(s_{c^*,j})$. Finally $A_{\text{PACK}}$ returns $\mathsf{OPT}(r)$. We assume that the DP solves the given instance optimally, i.e., $\mathsf{OPT}(r)$ is feasible and $\mathsf{opt}(r) = p(\mathsf{OPT}(r)) = \mathsf{opt}$, where $\mathsf{opt}$ denotes the value of an optimal solution to the given instance.

DPs with the above structure are very natural for packing problems, therefore, we refer to them as *packing DPs.* For example, the DP for Knapsack described above is of this type. There, the set of states $\mathcal{S}$ consists of all pairs $(i, C')$ with $i \in \{0, 1, ..., n\}$ and $C' \in \{0, 1, ..., C\}$, and the respective choices are whether to select $i$ or not. The base states $\mathcal{S}_{\text{base}}$ are the states $s = (i, C') \in \mathcal{S}$ with $i = 0$; for those, we naturally have $\mathsf{opt}(s) = 0$ and $\mathsf{OPT}(s) = \emptyset$. Note that the states and choices are oblivious to the item profits, as required.

**3 Algorithm for unary DPs and Cardinality Constraints** In this section we assume that for each state $s \in \mathcal{S} \setminus \mathcal{S}_{\text{base}}$ and each choice $c \in CH(s)$ we have that $d(c) = 1$, i.e., when making the choice $c$, there is only one single child state $s_{c,1}$. For convenience, we simply call this state $s_c$. We say that such a DP is an *unary DP.* For example, the mentioned packing DP for Knapsack is unary. For this setting, we present a simpler algorithm with a stronger guarantee than in the general setting; this will prove Theorem 1.2. We remark that our meta-algorithm for unary DPs is sufficient to obtain a PTAS for Knapsack with $O(\log n/\log\log\log n)$ additional (arbitrary) cardinality constraints as mentioned in Section 1.3. Previously this was not known for $\omega(1)$ such constraints. Furthermore, this meta-algorithm already uses some of the key ideas of our meta-algorithm of Theorem 1.1. We refer to the full version of this article for all the omitted proofs in this section.

In order to simplify the presentation, we will impose some extra properties on the input packing DP w.l.o.g. according to the following lemma.

LEMMA 3.1. *Suppose we are given a unary packing DP. By increasing the number of states and choices by at most a polynomial factor, we can assume that*

*(P1) for each state $s \in \mathcal{S} \setminus \mathcal{S}_{\text{base}}$ and each choice $c \in CH(s)$, either $I(c) = \emptyset$ or $I(c)$ contains exactly one item that we denote by $i(c)$,*

*(P2) for each state $s \in \mathcal{S} \setminus \mathcal{S}_{\text{base}}$ and each choice $c \in CH(s)$ with $I(c) \neq \emptyset$, there is another choice $c' \in CH(s)$ with $I(c') = \emptyset$ and $s_c = s_{c'}$, and*

*(P3) for each state $s \in \mathcal{S} \setminus \{r\}$ there is a state $s' \in \mathcal{S}$ and a choice $c \in CH(s')$ with $s = s_c$.*

Property (P2) states that for the each choice $c \in CH(s)$ there is a choice $c' \in CH(s)$ leading to the same subsequent state as $c$ but which does not select any items. This is justified since we assumed the set of solutions $\mathcal{F}$ to be downward closed. Property (P3) states that apart from the root state $r$, each state $s$ can be reached if we make a suitable choice in some other state $s'$. We can ensure it by simply removing the states that are not reachable from the root; this does not affect the final outcome of the DP.

A *choice vector* is a vector $\mathbf{c} = (c(s))_{s \in \mathcal{S} \setminus S_{\text{base}}}$ with a choice $c(s) \in CH(s)$ for each state $s \in \mathcal{S} \setminus S_{\text{base}}$. We define an associated *selection path* $P_{\mathbf{c}} = (V(P_{\mathbf{c}}), E(P_{\mathbf{c}}))$ as follows. We define that $V(P_{\mathbf{c}})$ contains the vertex $r$ corresponding to the root state. Recursively, for each vertex $s \in V(P_{\mathbf{c}})$ with $s \in \mathcal{S} \setminus \mathcal{S}_{\text{base}}$ we define that $V(P_{\mathbf{c}})$ contains also the vertex $s_{c(s)}$ and that $E(P_{\mathbf{c}})$ contains the arc $(s, s_{c(s)})$. For each $s \in V(P_{\mathbf{c}})$, we define $S_{\mathbf{c}}(s)$ to be the union of all sets $I(c(s'))$ of the descendants $s'$ of $s$ in $P_{\mathbf{c}}$ (including $s$), and we define $S_{\mathbf{c}} = S_{\mathbf{c}}(r)$ to be the *solution corresponding to* $\mathbf{c}$.

LEMMA 3.2. *Given a choice vector $\mathbf{c} = (c(s))_{s \in \mathcal{S} \setminus S_{\text{base}}}$, $P_{\mathbf{c}}$ is a directed path rooted at $r$. For any $s \in \mathcal{S} \setminus \mathcal{S}_{\text{base}}$, we have that $S_{\mathbf{c}}(s) = I(c(s)) \dot\cup S_{\mathbf{c}}(s_{c(s)})$.*

We say that a set $S$ is a *potential solution* if there is a choice vector $\mathbf{c} = (c(s))_{s \in \mathcal{S} \setminus S_{\text{base}}}$ such that $S = S_{\mathbf{c}}$. Exploiting the properties from Lemma 3.1, we can show that that the potential solutions of the DP are exactly $\mathcal{F}$, i.e., every feasible solution is also a potential solution and vice versa.

LEMMA 3.3. *The set of potential solutions equals $\mathcal{F}$.*

Recall that we consider the problem $\textsc{Pack}^+ := \textsc{Pack}_C^+$ obtained from $\textsc{Pack}$ by imposing $M$ additional cardinality constraints, requiring that any feasible solution $S$ satisfies that $|S \cap I_h| \le b_h$ for each $h \in [M]$. Let $\mathcal{F}^+$ be the set of all feasible solutions to a given instance of $\textsc{Pack}^+$. Obviously, $\mathcal{F}^+ \subseteq \mathcal{F}$ and $\mathcal{F}^+$ is downward closed. Recall that $\mathsf{OPT}^+$ denotes some optimal solution to the considered instance and that $\mathsf{opt}^+$ denotes its profit.

**3.1 A Slow Exact DP** In order to highlight the main ideas in our approach, it is convenient to first discuss a simple approach that allows one to solve $\textsc{Pack}^+$ optimally in polynomial time for $M = O(1)$. For the sake of simplicity we first consider the case $M = 1$.

We define a DP with a set of states $\mathcal{S}^+$. For each state $s \in \mathcal{S}$ and each $b \in \{0, \dots, b_1\}$ we create a state $(s, b) \in \mathcal{S}^+$. Intuitively, this state $(s, b)$ corresponds to being in the state $s \in \mathcal{S}$ such that we can still select $b$ items from $I_1$ in all states descending from $s$. For example, if $\mathcal{S}$ are the states of the DP for Knapsack described above, then each state in $\mathcal{S}^+$ corresponds to an item $i$, a residual capacity of the knapsack, and the constraint that at most $b$ selected items from $1, ..., i$ may be contained in $I_1$. If $s \in \mathcal{S}_{\text{base}}$, then we define that $(s, b)$ belongs to the set $\mathcal{S}^+_{\text{base}}$ of base states of our new DP. For each state $(s, b) \in \mathcal{S}^+$, we define a set of choices $CH^+(s, b)$ as follows. If $(s, b) \in \mathcal{S}^+_{\text{base}}$ then $CH^+(s, b) := \emptyset$. Assume now that $(s, b) \in \mathcal{S}^+ \setminus \mathcal{S}^+_{\text{base}}$; hence, $s \notin \mathcal{S}_{\text{base}}$. For each choice $c \in CH(s)$ we do the following. If $|I(c) \cap I_1| = |\{i(c)\}| = 1$ and $b = 0$, then, intuitively, there is not enough residual budget in the considered subproblem to add the item $i(c)$. Therefore, we ignore this choice $c$. (However, recall that we assumed that there exists another choice $c'$ that leads to the same state as $c$ but does not select any item, see Lemma 3.1.) Otherwise, we add to $CH^+(s, b)$ a choice $c^+$ (corresponding to $c$) for which we define $I(c^+) := I(c)$ and $s_{c^+} := (s_c, b - |I(c) \cap I_1|)$. Finally, we define the root state to be $(r, b_1)$.

Given these states and choices, we compute for each state $(s, b) \in \mathcal{S}^+$ a solution $\mathsf{OPT}^+(s, b)$ with a value $\mathsf{opt}^+(s, b)$ similarly as above in (2.1). Formally, for each state $(s, b) \in \mathcal{S}^+_{\text{base}}$ we define $\mathsf{OPT}^+(s, b) := \emptyset$ and $\mathsf{opt}^+(s, b) = 0$; for each state $(s, b) \in \mathcal{S}^+ \setminus \mathcal{S}^+_{\text{base}}$ we define

$$\mathsf{opt}^+(s, b) := \max_{c^+ \in CH^+(s,b)} \{p(I(c^+)) + \mathsf{opt}^+(s_{c^+})\} \tag{3.1}$$

and $\mathsf{OPT}^+(s, b) := I(c^*) \cup \mathsf{OPT}^+(s_{c^*})$ for a choice $c^* \in CH^+(s, b)$ that attains the maximum in (3.1). E.g., given the packing DP for Knapsack as discussed in Section 1, for any $i > 0$ one has $\mathsf{opt}^+((i, C'), b) = \mathsf{opt}^+((i-1, C'), b)$ if $i$ cannot be selected because $s(i) > C'$ or both $i \in I_1$ and $b = 0$. Otherwise $\mathsf{opt}^+((i, C'), b) = \max\{\mathsf{opt}^+((i - 1, C'), b), p(i) + \mathsf{opt}^+((i-1, C' - s(i)), b - |\{i\} \cap I_1|)\}$. We output the solution $\mathsf{OPT}^+(r, b_1)$ corresponding to the root state $(r, b_1)$ which turns out to be an optimal solution.

Lemma 3.4. *$\mathsf{OPT}^+(r, b_1)$ is an optimal solution for the given instance of $\textsc{Pack}^+$.*

Therefore, we can solve $\textsc{Pack}^+$ in a running time that is polynomial in $|\mathcal{S}^+|$, $\sum_{s \in \mathcal{S}} |CH(s)|$, and $b_1 \le n$. However, we observe that $|\mathcal{S}^+|$ can be by a factor $\Theta(n)$ larger than $|\mathcal{S}|$ (in particular, this happens for $b_1 = \Theta(n)$). This implies that the running time of our algorithm may be $\Theta(n)$ times larger than the running time of $A_{\textsc{Pack}}$.

The above construction can be naturally extended to any number $M$ of cardinality constraints. This leads to an increase of the worst-case running time by a factor $\Theta(n^M)$.

**3.2 A Fast Approximate Probabilistic DP** We next describe a much faster probabilistic DP which we will use to compute a $(1 + \varepsilon)$-approximate feasible solution. More specifically, the running time increases only by a factor $O(1/\delta)$ per cardinality constraint, where we choose $\delta = \Theta((\frac{\varepsilon}{\log(Mn)})^{O(1)})$ such that $1/\delta$ is an integer[3]. Hence, the overall running time increases only by a factor of $\left(\frac{\log(Mn)}{\varepsilon}\right)^{O(M)}$ while we lose only a small factor of $1 + \varepsilon$ in the obtained profit. Let $\varepsilon \le 1/4$ such that $1/\varepsilon$ is an integer. By standard rounding and scaling of the item profits, we assume that $p(i) \in [1, n/\varepsilon]$ for each $i \in I$, while losing at most a factor of $1 + \varepsilon$ in the approximation ratio. Again, for simplicity, we first consider the case where $M = 1$.

On a high level, the DP from Section 3.1 computes one choice $c^+$ for each given state $(s, b) \in \mathcal{S}^+ \setminus \mathcal{S}^+_{\text{base}}$ and this choice leads *deterministically* to a state $s_{c^+}$. The DP computes choices that maximize the total profit of making these choices (starting in the root $r$). Instead, we derive now a different DP that again computes one choice for each state in $\mathcal{S}^+ \setminus \mathcal{S}^+_{\text{base}}$; however, in some cases the child state is chosen *randomly* according to some given probability distribution. More precisely, the child state will be of the form $(s_c, B)$ for some $s_c \in \mathcal{S}$ where

[3]In the unary case a larger value of $\delta$ can be used, which leads to a better running time as claimed in Theorem 1.2. However, for simplicity we use here the same value as in the general case of not necessarily unary DPs.

$B$ is a *random variable.* Therefore, our new DP selects choices that maximize the *expected* profit of this random process (starting in the root $r$). Nevertheless, the DP itself is still a *deterministic* algorithm. In a post-processing step, we will then use a *randomized* algorithm to translate the choices computed by the DP to an actual set of items, which we output at the end. An important technical ingredient in our construction is a careful restriction of the allowed budgets.

Formally, in the case that $b_1 \leq 1/\delta$ we can use the approach from Section 3.1 since the increase of the running time is only by a factor of $O(1/\delta)$ as desired. Hence, we assume now that $b_1 > 1/\delta$. We restrict the budgets $b$ for the states $(s,b) \in \mathcal{S}^+$ to the set $\mathcal{B} := \{k \cdot \delta \cdot b_1 : 0 \leq k \leq 1/\delta\}$. This yields a reduced set of states $\mathcal{S}^+ := \{(s,b) : s \in \mathcal{S} \wedge b \in \mathcal{B}\}$. The root state is $(r, b_1)$. For each base state $(s,b) \in \mathcal{S}^+_{\text{base}}$ we define $CH(s,b) := \emptyset$. For each non-base state $(s,b) \in \mathcal{S}^+ \setminus \mathcal{S}^+_{\text{base}}$ we define a set of choices $CH^+(s,b)$ as follows. Consider a choice $c \in CH(s)$. If $|I(c) \cap I_1| = |\{i(c)\}| = 1$ and $b = 0$, then we do not construct any choice that corresponds to $c$ since, intuitively, there is no budget left to select $I(c)$. Otherwise we construct a corresponding choice $c^+ \in CH^+(s,b)$ by setting $I(c^+) := I(c)$ and defining up to two possible child states, each of them with a certain probability. If $I(c) \cap I_1 = \emptyset$ then there is only one child state $(s_c, b)$ which is selected deterministically. If $I(c) \cap I_1 \neq \emptyset$ there are two child states: the state $(s_c, b - \delta \cdot b_1)$[4] which is selected with probability $\frac{1}{\delta \cdot b_1}$ and the child state $(s_c, b)$ which is selected otherwise. Hence, the child state is of the form $(s_c, B)$ where $B$ is a random variable with $\mathbb{E}[B] = b - |I(c) \cap I_1|$ while deterministically $B \geq 0$. This defines our choice $c^+ \in CH^+(s,b)$ corresponding to $c$. We do this proceduce for each state $(s,b)$ and each choice $c \in CH(s)$. Note that the states and choices above correspond to a randomized decision process. In each state, we need to make a choice. Then, *after we selected the choice*, the child state may be defined randomly.

Similar as in the DP for Pack, our goal is to compute a choice vector $\mathbf{c} = (c(s,b))_{(s,b) \in \mathcal{S}^+ \setminus \mathcal{S}^+_{\text{base}}}$ which now defines a *random* selection path $P_{\mathbf{c}} = (V(P_{\mathbf{c}}), E(P_{\mathbf{c}}))$ defined as follows. We start by adding the vertex $(r, b_1)$ corresponding to the root state to $V(P_{\mathbf{c}})$ and initialize $S_{\mathbf{c}} := \emptyset$. Recursively, assuming that $(s,b) \in V(P_{\mathbf{c}})$ was the last vertex added to $P_{\mathbf{c}}$, if $(s,b) \in \mathcal{S}^+_{\text{base}}$ we stop the recursion. Otherwise, we proceed according to the choice $c(s,b)$: if $I(c(s,b)) \cap I_1 = \emptyset$ then we define deterministically $B := b$; if $I(c(s,b)) \cap I_1 \neq \emptyset$ we define $B := b - \delta \cdot b_1$ with probability $\frac{1}{\delta \cdot b_1}$ and $B := b$ otherwise. We add the items in $I(c(s,b))$ to $S_{\mathbf{c}}$, add the vertex $(s_c, B)$ to $V(P_{\mathbf{c}})$, add the arc $((s,b),(s_c,B))$ to $E(P_{\mathbf{c}})$, and continue recursively with the state $(s_c, B)$. Hence, both $P_{\mathbf{c}}$ and $S_{\mathbf{c}}$ are random objects.

We define a DP computing a choice vector $\mathbf{c}$ that maximizes the expected profit of the set $S_{\mathbf{c}}$. For each state $(s,b) \in \mathcal{S}^+$, our DP computes a value $\mathsf{apx}^+(s,b)$ which is the expected profit from the state $(s,b)$ and all its child states if in each of these states we make the choice that maximizes the expected profit. Formally, for each state $(s,b) \in \mathcal{S}^+$ we set $\mathsf{apx}^+(s,b) := 0$ if $(s,b) \in \mathcal{S}^+_{\text{base}}$ and otherwise we define

$$\begin{aligned} \mathsf{apx}^+(s,b) := \max_{c^+ \in CH^+(s,b)} \{ & p(I(c^+)) + \mathbb{P}[B = b | c^+] \cdot \mathsf{apx}^+(s_c, b) \\ & + \mathbb{P}[B = b - b_1 \cdot \delta | c^+] \cdot \mathsf{apx}^+(s_c, b - b_1 \cdot \delta)\}. \end{aligned} \tag{3.2}$$

For example, in the case of Knapsack, for each $i \geq 1$ the value $\mathsf{apx}^+((i, C'), b)$ is defined as follows if we can select item $i$, i.e., if $s(i) \leq C'$ and additionally $b_1 \geq 1$ or $i \notin I_1$:

$$\begin{aligned} \mathsf{apx}^+((i,C'),b) := \max\{ & \mathsf{apx}^+((i-1,C'),b), \\ & p(i) + \frac{1}{\delta b_1} \mathsf{apx}^+((i-1, C' - s(i)), b - \delta\, b_1) + \big(1 - \frac{1}{\delta b_1}\big) \mathsf{apx}^+((i-1, C' - s(i)), b)\}. \end{aligned}$$

Otherwise, $\mathsf{apx}^+((i,C'),b)$ is defined analogously to $\mathsf{opt}^+((i,C'),b)$ (as in Section 3.1). Like in Section 3.1, for each state $(s,b) \in \mathcal{S}^+$, we also store the choice $c^+ \in CH^+(s,b)$ that yields the maximum in (3.2).

We want to show that for the root state $(r, b_1)$ the computed value $\mathsf{apx}^+(r, b_1)$ is essentially $\mathsf{opt}^+$. In order to prove this, we show that there exists a choice vector $\mathbf{c}$ via which we obtain a profit of at least $(1 - O(\varepsilon))\mathsf{opt}^+$ in expectation. This suffices since our DP computes a choice vector that maximizes the expected profit. We define $\mathbf{c}$ as follows, using the solution $\mathsf{OPT}'$ from the following lemma.

Lemma 3.5. *Assume that $b_1 \geq 1/\delta \geq 12 \log n / \varepsilon^3$. Then there exists a solution $\mathsf{OPT}'$ with $|\mathsf{OPT}' \cap I_1| \leq (1-\varepsilon) b_1$ and $p(\mathsf{OPT}') \geq (1 - 3\varepsilon)\mathsf{opt}^+$.*

[4]Observe that $b > 0$ implies $b \geq \delta\, b_1$.

*Proof sketch.* Let $\mathsf{OPT}'$ be obtained by removing every item in $\mathsf{OPT}^+$ with probability $2\varepsilon$. Standard Chernoff bounds [46] show that w.h.p. $|\mathsf{OPT}' \cap I_1| \le (1-\varepsilon)b_1$. As the event $|\mathsf{OPT}' \cap I_1| > (1-\varepsilon)b_1$ is very unlikely, at least $(1-3\varepsilon)\mathsf{opt}^+$ of the expected profit of $\mathsf{OPT}'$ must stem from the case where $|\mathsf{OPT}' \cap I_1| \le (1-\varepsilon)b_1$. Therefore there must exist a set $\mathsf{OPT}'$ with the properties claimed by the Lemma. ∎

Due to Lemma 3.3, there is a choice vector $\bar{\mathbf{c}} = (\bar{c}(s))_{s\in\mathcal{S}}$ for the DP for Pack that constructs $\mathsf{OPT}'$. We define that our choice vector $\mathbf{c}$ simply makes the same decisions as $\bar{\mathbf{c}}$, i.e., $c(s,b) = \bar{c}(s)$ for each state $(s,b) \in \mathcal{S}^+ \setminus \mathcal{S}^+_{\text{base}}$. The only exception is when we are in a state $(s,b)$ with $b = 0$ and the decision $\bar{c}(s)$ is not available (due to lack of budget). Then we simply make the decision that does not select any item but still leads to the same child state $s_{\bar{c}(s)}$. This decision is available due to Property (P2) of Lemma 3.3. Now we can consider the (random) solution $S_{\bar{\mathbf{c}}}$. We will show that $S_{\bar{\mathbf{c}}} = \mathsf{OPT}'$ with probability at least $1-\varepsilon$. As $\mathsf{apx}^+(r,b_1) \ge \mathbb{E}[p(S_{\bar{\mathbf{c}}})]$ this yields the following lemma.

Lemma 3.6. *We have that* $\mathsf{apx}^+(r,b_1) \ge (1-O(\varepsilon))\mathsf{opt}^+$.

*Proof sketch.* One can show that $\mathsf{apx}^+(r,b_1)$ is indeed the expected profit obtained by an (optimal) choice vector that maximizes the expected profit. When we construct the random solution $S_{\bar{\mathbf{c}}}$ corresponding to the choice vector $\bar{\mathbf{c}}$, the remaining budget decreases by a random value $Z_k$ with $\mathbb{P}[Z_k = \delta b_1] = \frac{1}{\delta b_1}$ and $\mathbb{P}[Z_k = 0] = 1 - \frac{1}{\delta b_1}$ each time we select an item from $\mathsf{OPT}' \cap I_1$. If the remaining budget is not 0 at the end, then it was positive at each intermediate step. Thus, we could select each item from $\mathsf{OPT}'$. Using Chernoff bound on the values $Z_k$, we can show that this happens with probability at least $1-\varepsilon$. Therefore, the expected profit of the sampled solution is at least $(1-O(\varepsilon))\mathsf{opt}^+$. ∎

**3.3 Rounding of the approximate DP** Let $\mathbf{c}^* = (c^*(s,b))_{(s,b)\in\mathcal{S}^+\setminus\mathcal{S}^+_{\text{base}}}$ be the choice vector computed by the above DP. We next show how to derive a feasible solution of large expected profit from $\mathbf{c}^*$. First, we sample a random selection path $P_{\mathbf{c}^*}$ and the corresponding set $S_{\mathbf{c}^*} =: \mathsf{APX}^+$ from $\mathbf{c}^*$ as described above. One can easily show that $\mathbb{E}[p(\mathsf{APX}^+)] = \mathsf{apx}^+(r,b_1)$. However, $\mathsf{APX}^+$ is not necessarily a feasible solution to Pack$^+$. More precisely, deterministically $\mathsf{APX}^+ \in \mathcal{F}$, but it might happen that $|\mathsf{APX}^+ \cap I_1| > b_1$. The reason is that when we define $\mathsf{APX}^+$ there may be many states $(s,b)$ in which $c^*(s,b)$ selects an item from $I_1$, but in the child state $(s_{c^*(s,b)}, B)$ we still have that $B = b$, i.e., the remaining budget is not decreased even though we selected an item from $I_1$. However, with the Chernoff bound we can show that w.h.p. the cardinality constraint due to $I_1$ is exceeded by at most a small amount. Altogether, we obtain the following lemma.

Lemma 3.7. *One has: (1) deterministically* $\mathsf{APX}^+ \in \mathcal{F}$*; (2)* $|\mathsf{APX}^+ \cap I_1| \le (1+\varepsilon)b_1$ *with probability at least* $1-\frac{\varepsilon}{n}$*; (3)* $\mathbb{E}[p(\mathsf{APX}^+)] = \mathsf{apx}^+(r,b_1)$.

It is now easy to derive from $\mathsf{APX}^+$ a feasible solution $\widetilde{\mathsf{APX}}$ of large expected profit: drop each item in $\mathsf{APX}^+$ independently with probability $2\varepsilon$. Then in expectation we keep a fraction $1-2\varepsilon$ of the original profit and w.h.p. $\widetilde{\mathsf{APX}}$ is feasible (otherwise, set $\widetilde{\mathsf{APX}} = \emptyset$). Since our item profits are in a polynomial range, this guarantees that $\mathbb{E}[p(\widetilde{\mathsf{APX}})] \ge (1-O(\varepsilon))\mathbb{E}[p(\mathsf{APX}^+)] = (1-O(\varepsilon))\mathsf{apx}^+(r,b_1') \ge (1-O(\varepsilon))\mathsf{opt}^+$.

This yields a PTAS for Pack$^+$ for the case that $M = 1$. We generalize the approach above to arbitrary values of $M$ as follows. For each cardinality constraint $h$ for which $b_h \le 1/\delta$, we use the approach from Section 3.1. Therefore, let us assume that $b_h > 1/\delta$ for each cardinality constraint $h$. Each state $s^+ \in \mathcal{S}^+$ is of the form $(s,\mathbf{b})$ where $s \in \mathcal{S}$ and $\mathbf{b} = (b^{(1)}, b^{(2)}, ..., b^{(M)})$ such that for each $h \in [M]$ the entry $b^{(h)}$ satisfies $b^{(h)} \in \{k \cdot \delta \cdot b_h : 0 \le k \le 1/\delta\}$, denoting the remaining budget for the $h$-th cardinality constraint in the state and all its descendant states. Thus, for each state $s \in \mathcal{S}$ there are $(1/\delta)^{O(M)}$ states $s^+$ in $\mathcal{S}^+$. Given one state $s \in \mathcal{S}$, for each choice $c \in CH(s)$ we introduce $(1/\delta)^{O(M)}$ different choices for $s^+$. Each such choice defines, for each cardinality constraint $h$, how the $h$-th budget $b^{(h)}$ is adjusted to a (possibly random) budget $B^{(h)}$ of the child state. We can prove analogues of Lemmas 3.6 and 3.7 where in the latter lemma property (2) then states that $|\mathsf{APX}^+ \cap I_h| \le (1+\varepsilon)b_h$ for every $h = 1, \dots, M$ with sufficiently high probability. Finally, we drop items randomly similarly to the case $M = 1$. We omit the details here; this yields Theorem 1.2.

**4 Hardness of approximation** In this section, we prove Theorem 1.3 and Theorem 1.4 by reducing 3-SAT and the $d$-dimensional Vector Knapsack ($d$-VK) problem to two simple packing problems (with simple packing DPs) and additional cardinality and budget constraints, respectively. Then, our hardness results hold assuming the Randomized Gap-ETH, as it is for example formulated in [10] (based on the Gap-ETH, see [14, 42]).

*Conjecture* 4.1 (Randomized Gap-ETH). For some constants $\delta, \varepsilon > 0$, no algorithm can, given a 3-SAT formula $\phi$ on $n$ variables and $m = O(n)$ clauses, distinguish between the following cases correctly with probability $\geq 2/3$ in $O(2^{\delta n})$ time:

- $\mathrm{SAT}(\phi) = m$ and
- $\mathrm{SAT}(\phi) < (1-\varepsilon)m$,

where $\mathrm{SAT}(\phi)$ is the maximum number of clauses in $\phi$ that can be simultaneously satisfied.

*Proof of Theorem 1.3.* We consider a problem Pack defined as follows: an instance of Pack is a set of $n$ items $I$ and $2K$ disjoint subsets $V_1^0, \ldots, V_K^0, V_1^1, \ldots, V_K^1$ of $I$. The goal is find a maximum subset $I' \subseteq I$ such that $I' \cap V_k^0 = \emptyset$ or $I' \cap V_k^1 = \emptyset$ (or both) for every $k \in [K]$. No item can be taken from $I \setminus \left(\bigcup_{j \in \{0,1\}, k \in [K]} V_k^j\right)$.

Trivially, the optimal solution is to either take all items in $V_k^0$ if $|V_k^0| \geq |V_k^1|$, or all items in $V_k^1$ otherwise, for every $k \in [K]$. We reformulate this as a dynamic program: The states in $\mathcal{S}$ are $r$ (the root state) and states $s_{j,k}$ with $j \in \{0,1\}$ and $k \in [K] \cup \{0\}$ (where $s_{j,0}$ are base states). The state $s_{j,k}$ corresponds to the process of picking the items in $V_k^j$ (instead of the items in $V_k^{\neg j}$). For every $k \in [K]$, $j \in \{0,1\}$, $CH(s_{j,k})$ contains two choices $c_k^{j,j'}$, $j' \in \{0,1\}$ with child state $s_{j',k-1}$ and $I(c_k^{j,j'}) = V_k^j$. The choice $c_k^{j,j'}$ corresponds to the decision of picking the items in $V_k^j$ while the items in $V_{k-1}^{j'}$ where chosen before (assuming that we choose items in increasing index $k \in [K]$). It is also straightforward to verify that this solves the problem and the DP fulfills all our necessary criteria. Even after the preprocessing step in Lemma 3.1, the size and running time of the DP is clearly bounded by $(n+K)^{O(1)}$.

Let $f : \mathbb{N} \to \mathbb{N}$ be a monotonic function in $\omega(\log n)$. Assume now for the sake of contradiction that there exists a PTAS for Pack$^+$ with $f(n)$ additional disjoint cardinality constraints (with budget 1). We now argue that the MAX-3-SAT problem can be reduced to Pack$^+$ and deduce that the existence of a PTAS contradicts the Gap-ETH.

For a given 3-SAT instance $\phi = \bigwedge_{m \in [M]} (\ell_1^m \vee \ell_2^m \vee \ell_3^m)$ on $K$ variables $\{x_k\}_{k \in K}$ and $M$ clauses, we define an instance of Pack$^+$ with a set of items $I_\phi = \{\ell_i^m\}_{m \in [M], i \in [3]}$. For every $k \in K$, we create the pair of sets $V_k^1 = \{\ell_i^m \in I_\phi : \ \ell_i^m = x_k\}$ and $V_k^0 = \{\ell_i^m \in I_\phi : \ \ell_i^m = \neg x_k\}$. For every clause $C_m = \{\ell_1^m, \ell_2^m, \ell_3^m\}$, we impose a cardinality constraint with budget 1. A Pack$^+$ feasible solution $I' \subseteq I_\phi$ induces a assignment of $\phi$ such that at least $|I'|$ clauses in $\phi$ are satisfied: the assignment is defined as $x_k = $ false if $I' \cap V_k^0 \neq \emptyset$ and $x_k = $ true otherwise, e.g., if $I' \cap V_k^1 \neq \emptyset$. Therefore, if $\ell_i^m \in I'$, then the clause $(\ell_1^m \vee \ell_2^m \vee \ell_3^m)$ in $\phi$ is satisfied (and any literal cannot be both true and false by the constraint of Pack). Furthermore, by the cardinality constraints, for every satisfied clause $I_m$, at most one item in $I'$ can certify the satisfaction of the clause, so at least $|I'|$ clauses are satisfied. Note that *exactly* $|I'|$ clauses are satisfied if and only if $I'$ is a maximal Pack$^+$-feasible solution, i.e., if adding any item to $I'$ makes it Pack$^+$-infeasible.

Consider now a 3-SAT instance $\phi$ on $K$ variables and $M = O(K)$ clauses as in Conjecture 4.1 and consider its Pack$^+$-reduction $I_\phi$. We add dummy items (i.e., that do not belong to any $V_k^j$) to $I_\phi$ such that the resulting set $\tilde{I}_\phi$ contains $M' = f^{-1}(M)$ items. Since $f(n) = \omega(\log n)$ and $K = \Omega(M)$, we have

$$\frac{\log M'}{K} = \frac{M}{K} \cdot \frac{\log(f^{-1}(M))}{f(f^{-1}(M))} \to 0,$$

so $\log M' = o(K)$ and thus $M' = 2^{o(K)}$. The Pack$^+$ instance $\tilde{I}_\phi$ has therefore cardinality $M'$ and $f(M') = \omega(\log M')$ cardinality constraints. Applying the PTAS on the instance $\tilde{I}_\phi$, which has therefore running time $M'^{O(1)} = 2^{o(K)}$ allows us to make the distinction in Conjecture 4.1, which is a contradiction. ◻

*Remark* 4.2. Assuming the Exponential Time Hypothesis [31] and the Sparsification Lemma [32] instead of the Gap-ETH, it is impossible to solve the problem *exactly* for $\Theta(\log n)$ cardinality constraints, but this does not rule out a polynomial-time approximation algorithm for this case.

We prove Theorem 1.4 using the following result.

Theorem 4.3 ([15]). *Assuming the (randomized) Gap-ETH, for any constant $\varepsilon > 0$, there exist constants $\delta > 0$ and $d_0 \in \mathbb{N}$ such that the following holds: for any constant $d \geq d_0$, there is no (randomized) $\rho$-approximation algorithm for $d$-dimensional Vector Knapsack that runs in time $O(n^{\delta\sqrt{d}})$, even in the cardinality case.*

*Proof of Theorem 1.4.* We define PACK as the "trivial" Knapsack problem, namely where each item has weight 0 and the Knapsack has capacity 0, and the DP is the standard packing DP from Section 1. Obviously, the DP picks every item.

By adding $M$ budget constraints, we can end up with an arbitrary instance of $M$-dimensional Vector Knapsack. By Theorem 4.3, assuming the (Randomized) Gap-ETH, there is no (randomized) $(1+\varepsilon)$-approximation algorithm for $M$-VK with running time $O\big(n^{\delta\sqrt{M}}\big)$ (for $M \geq d_0$), where $\varepsilon$ is considered constant and $\delta > 0$ and $d_0 \in \mathbb{N}$ is another constant depending on $\varepsilon$. The claim follows. ∎

*Remark* 4.4. Strictly speaking, The authors of [15] prove the deterministic variant of Theorem 4.3, that is, that the deterministic Gap-ETH (i.e., with probability 1 instead of $\geq 2/3$ in the statement of Conjecture 4.1) implies that there is no *deterministic* $\rho$-approximation algorithm for $d$-VK, see [15, Theorem 1.4]. However, it is straightforward to adapt the proofs in [15] that yield to the latter theorem: The first step in [15] is to reduce 3-SAT to the so-called R-CSP problem (a variant of the Constraint Satisfaction Problem as defined in [15, Definition 2.1]) like in [15, Appendix A.2], and then to reduce R-CSP to $d$-VK, see [15, Lemma 3.1]. It can easily be observed that in the latter reduction, all the profits of the items are 1. Both reductions are constructive and deterministic, so the probability of successfully distinguishing an optimal solution and a $(1-\varepsilon)$-suboptimal solution like in Conjecture 4.1 is preserved in both reductions, so the existence of a *randomized* PTAS for $M$-VK (for $M = \omega(1)$) would contradict the *Randomized* Gap-ETH.

**Acknowledgments.** The authors thank the anonymous reviewers for their helpful comments.

## References

[1] A. ABDULKADIROĞLU AND T. SÖNMEZ, *School choice: A mechanism design approach*, American economic review, 93 (2003), pp. 729–747.

[2] F. ABED, P. CHALERMSOOK, J. CORREA, A. KARRENBAUER, P. PÉREZ-LANTERO, J. A. SOTO, AND A. WIESE, *On guillotine cutting sequences*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM), 2015, pp. 1–19.

[3] A. ANAGNOSTOPOULOS, F. GRANDONI, S. LEONARDI, AND A. WIESE, *Constant integrality gap LP formulations of unsplittable flow on a path*, in International Conference on Integer Programming and Combinatorial Optimization (IPCO), 2013, pp. 25–36.

[4] E. M. ARKIN AND E. B. SILVERBERG, *Scheduling jobs with fixed start and end times*, Discrete Applied Mathematics, 18 (1987), pp. 1–8.

[5] M. BANSAL AND V. VENKAIAH, *Improved fully polynomial time approximation scheme for the 0-1 multiple-choice knapsack problem*, International Institute of Information Technology Tech Report, (2004), pp. 1–9.

[6] R. BELMONTE, M. LAMPIS, AND V. MITSOU, *Parameterized (approximate) defective coloring*, SIAM Journal on Discrete Mathematics, 34 (2020), pp. 1084–1106.

[7] A. BERGER, V. BONIFACI, F. GRANDONI, AND G. SCHÄFER, *Budgeted matching and budgeted matroid intersection via the gasoline puzzle*, Mathematical Programming, 128 (2011), pp. 355–372.

[8] C. CALVO, C. IVORRA, AND V. LIERN, *Fuzzy portfolio selection including cardinality constraints and integer conditions*, Journal of Optimization Theory and Applications, 170 (2016), pp. 343–355.

[9] A. CAPRARA, H. KELLERER, U. PFERSCHY, AND D. PISINGER, *Approximation algorithms for knapsack problems with cardinality constraints*, European Journal of Operational Research, 123 (2000), pp. 333–345.

[10] P. CHALERMSOOK, M. CYGAN, G. KORTSARZ, B. LAEKHANUKIT, P. MANURANGSI, D. NANONGKAI, AND L. TREVISAN, *From gap-exponential time hypothesis to fixed parameter tractable inapproximability: Clique, dominating set, and more*, SIAM Journal on Computing, 49 (2020), pp. 772–810.

[11] C. CHEKURI AND S. KHANNA, *A polynomial time approximation scheme for the multiple knapsack problem*, SIAM Journal on Computing, 35 (2005), pp. 713–728.

[12] C. Chekuri, J. Vondrák, and R. Zenklusen, *Multi-budgeted matchings and matroid intersection via dependent rounding*, in Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms (SODA), 2011, pp. 1080–1097.

[13] G. Cormode and M. Hadjieleftheriou, *Methods for finding frequent items in data streams*, International Joournal on Very Large Data Bases, 19 (2010), pp. 3–20.

[14] I. Dinur, *Mildly exponential reduction from gap 3SAT to polynomial-gap label-cover*, in Electronic Colloquium on Computational Complexity (ECCC), 2016.

[15] I. Doron-Arad, A. Kulik, and P. Manurangsi, *Fine grained lower bounds for multidimensional knapsack*, arXiv preprint arXiv:2407.10146, (2024).

[16] I. Doron-Arad, A. Kulik, and H. Shachnai, *An EPTAS for budgeted matching and budgeted matroid intersection via representative sets*, in 50th International Colloquium on Automata, Languages, and Programming (ICALP), 2023, pp. 49:1–49:16.

[17] I. Doron-Arad, A. Kulik, and H. Shachnai, *An EPTAS for budgeted matroid independent set*, in Symposium on Simplicity in Algorithms (SOSA), 2023, pp. 69–83.

[18] I. Doron-Arad, A. Kulik, and H. Shachnai, *An FPTAS for budgeted laminar matroid independent set*, Operations Research Letters, 51 (2023), pp. 632–637.

[19] I. Doron-Arad, A. Kulik, and H. Shachnai, *Tight lower bounds for weighted matroid problems*, arXiv preprint arXiv:2307.07773, (2023).

[20] S. Duppala, J. Luque, J. Dickerson, and A. Srinivasan, *Group fairness in set packing problems*, in International Joint Conference on Artificial Intelligence (IJCAI), IJCAI, 2023.

[21] T. Fleiner and N. Kamiyama, *A matroid approach to stable matchings with lower quotas*, Mathematics of Operations Research, 41 (2016), pp. 734–744.

[22] W. Gálvez, F. Grandoni, S. Ingala, S. Heydrich, A. Khan, and A. Wiese, *Approximating geometric knapsack via L-packings*, ACM Transactions on Algorithms (TALG), 17 (2021), pp. 1–67.

[23] W. Gálvez, F. Grandoni, A. Khan, D. Ramírez-Romero, and A. Wiese, *Improved approximation algorithms for 2-dimensional knapsack: Packing into multiple L-shapes, spirals, and more*, in 37th International Symposium on Computational Geometry (SoCG), 2021, pp. 39:1–39:17.

[24] W. Gálvez, A. Khan, M. Mari, T. Mömke, M. R. Pittu, and A. Wiese, *A 3-approximation algorithm for maximum independent set of rectangles*, in Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2022, pp. 894–905.

[25] W. Gálvez, A. Khan, M. Mari, T. Mömke, M. Reddy, and A. Wiese, *A $(2+\varepsilon)$-approximation algorithm for maximum independent set of rectangles*, arXiv preprint arXiv:2106.00623, (2021).

[26] F. Grandoni, E. Husić, M. Mari, and A. Tinguely, *Approximating the maximum independent set of convex polygons with a bounded number of directions*, in 40th International Symposium on Computational Geometry (SoCG), 2024, pp. 61.1–61.16.

[27] F. Grandoni, T. Mömke, and A. Wiese, *A PTAS for unsplittable flow on a path*, in Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC), 2022, pp. 289–302.

[28] F. Grandoni and R. Zenklusen, *Approximation schemes for multi-budgeted independence systems*, in European Symposium on Algorithms (ESA), 2010, pp. 536–548.

[29] S. Heydrich and A. Wiese, *Faster approximation schemes for the two-dimensional knapsack problem*, ACM Transactions on Algorithms (TALG), 15 (2019), pp. 1–28.

[30] O. H. Ibarra and C. E. Kim, *Fast approximation algorithms for the knapsack and sum of subset problems*, Journal of the ACM (JACM), 22 (1975), pp. 463–468.

[31] R. Impagliazzo and R. Paturi, *On the complexity of k-SAT*, Journal of Computer and System Sciences, 62 (2001), pp. 367–375.

[32] R. Impagliazzo, R. Paturi, and F. Zane, *Which problems have strongly exponential complexity?*, Journal of Computer and System Sciences, 63 (2001), pp. 512–530.

[33] T. Inamdar, D. Lokshtanov, S. Saurabh, and V. Surianarayanan, *Parameterized complexity of fair bisection: FPT-approximation meets unbreakability*, arXiv preprint arXiv:2308.10657, (2023).

[34] K. Jansen, *Parameterized approximation scheme for the multiple knapsack problem*, SIAM Journal on Computing, 39 (2010), pp. 1392–1412.

[35] K. Jansen, *A fast approximation scheme for the multiple knapsack problem*, in International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), 2012, pp. 313–324.

[36] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack problems*, Springer, 2004.

[37] A. Khan, A. Maiti, A. Sharma, and A. Wiese, *On guillotine separable packings for the two-dimensional geometric knapsack problem*, in 37th International Symposium on Computational Geometry (SoCG), 2021, pp. 1–17.

[38] A. Khan and M. R. Pittu, *On guillotine separability of squares and rectangles*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM), 2020, pp. 47–1.

[39] M. Lampis, *Parameterized approximation schemes using graph widths*, in International Colloquium on Automata, Languages, and Programming (ICALP), 2014, pp. 775–786.

[40] E. L. Lawler, *Fast approximation algorithms for knapsack problems*, in 18th Annual Symposium on Foundations of Computer Science (SFCS), 1977, pp. 206–213.

[41] W. Li, J. Lee, and N. Shroff, *A faster FPTAS for knapsack problem with cardinality constraint*, Discrete Applied Mathematics, 315 (2022), pp. 71–85.

[42] P. Manurangsi and P. Raghavendra, *A birthday repetition theorem and complexity of approximating dense CSPs*, in 44th International Colloquium on Automata, Languages, and Programming (ICALP), 2017, pp. 78–1.

[43] R. K. Martin, R. L. Rardin, and B. A. Campbell, *Polyhedral characterization of discrete dynamic programming*, Operations research, 38 (1990), pp. 127–138.

[44] M. Mastrolilli and M. Hutter, *Hybrid rounding techniques for knapsack problems*, Discrete applied mathematics, 154 (2006), pp. 640–649.

[45] J. S. Mitchell, *Approximating maximum independent set for rectangles in the plane*, in 2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS), 2022, pp. 339–350.

[46] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*, Cambridge university press, 2017.

[47] K. Pruhs and G. J. Woeginger, *Approximation schemes for a class of subset selection problems*, Theoretical Computer Science, 382 (2007), pp. 151–156.

[48] G. S. Sankar, A. Louis, M. Nasre, and P. Nimbhorkar, *Matchings with group fairness constraints: Online and offline algorithms*, in Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI), 2021, pp. 377–383.

[49] P. Sinha and A. A. Zoltners, *The multiple-choice knapsack problem*, Operations Research, 27 (1979), pp. 503–515.

[50] V. V. Vazirani, *Approximation algorithms*, vol. 1, Springer, 2001.

[51] A. Wiese, *Independent set of convex polygons: From $n^\epsilon$ to $1+\epsilon$ via shrinking*, Algorithmica, 80 (2018), pp. 918–934.

[52] G. J. Woeginger, *When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)?*, INFORMS Journal on Computing, 12 (2000), pp. 57–74.