An AFPTAS for Bin Packing with Partition Matroid via a New Method for LP Rounding

Ilan Doron-Arad
Computer Science Department, Technion, Haifa, Israel

Ariel Kulik
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Hadas Shachnai
Computer Science Department, Technion, Haifa, Israel

Abstract

We consider the Bin Packing problem with a partition matroid constraint. The input is a set of items of sizes in $[0, 1]$, and a partition matroid over the items. The goal is to pack the items in a minimum number of unit-size bins, such that each bin forms an independent set in the matroid. This variant of classic Bin Packing has natural applications in secure storage on the Cloud, as well as in equitable scheduling and clustering with fairness constraints.

Our main result is an *asymptotic* fully *polynomial-time approximation scheme (AFPTAS)* for Bin Packing with a partition matroid constraint. This scheme generalizes the known AFPTAS for Bin Packing with Cardinality Constraints and improves the existing *asymptotic polynomial-time approximation scheme (APTAS)* for Group Bin Packing, which are both special cases of Bin Packing with partition matroid. We derive the scheme via a new method for rounding a (fractional) solution for a configuration-LP. Our method uses this solution to obtain *prototypes*, in which items are interpreted as placeholders for other items, and applies *fractional grouping* to modify a fractional solution (prototype) into one having desired integrality properties.

2012 ACM Subject Classification Theory of computation

Keywords and phrases bin packing, partition-matroid, AFPTAS, LP-rounding

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.22

Category APPROX

Related Version *Full Version*: `https://arxiv.org/abs/2212.01025` [9]

Funding *Ariel Kulik*: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 852780-ERC (SUBMODULAR).

1 Introduction

The *bin packing (BP)* problem involves packing a set of items in a minimum number of containers (bins) of the same (unit) size. Bin Packing is one of the most studied problems in combinatorial optimization. Indeed, in many real-life scenarios, a solution for BP is essential for optimizing the allocation of resources. In this paper, we consider the Bin Packing problem with a partition matroid constraint. The input is a set of items of sizes in $[0, 1]$, and a partition matroid over the items. The goal is to pack all the items in a minimum number of unit-size bins, such that each bin forms an independent set in the matroid.

Formally, a *bin packing with partition matroid (BPP)* instance is a tuple $\mathcal{I} = (I, \mathcal{G}, s, k)$, where I is a set of items, $\mathcal{G}$ is a partition of I into groups, $s : I \to [0, 1]$ gives the item sizes, and $k : \mathcal{G} \to \mathbb{N}_{>0}$ sets a cardinality constraint for each group. The *instance matroid* of $\mathcal{I}$ is the partition matroid $\mathcal{M} = (I, \mathcal{S})$ where $\mathcal{S} = \{S \subseteq I \mid \forall G \in \mathcal{G} : \ |S \cap G| \leq k(G)\}$.

A *configuration* of the instance $\mathcal{I}$ is a subset of items $C \subseteq I$ such that $\sum_{\ell \in C} s(\ell) \leq 1$ and $C \in \mathcal{S}$. That is, the total size of items in C is at most one, and for each *group* $G \in \mathcal{G}$ the configuration C contains at most $k(G)$ items from G. A *packing* of $\mathcal{I}$ is a partition of I into m subsets called *bins* $(A_1, \ldots, A_m)$ such that A_b is a configuration for all $b \in [m]$.[1] The objective is to find a packing of all items in a minimal number of bins. Indeed, the special case where each group consists of a *single* item and $k(G) = 1$ for all $G \in \mathcal{G}$ is the classic Bin Packing problem.

Bin Packing with Partition Matroid has natural application in secure storage of project data in the Cloud. Computational projects of large data scale often rely on cloud computing. Commonly, the project data is also stored in the cloud. In this setting, a main concern is that a malicious entity might gain access to confidential data [13]. To strengthen security, data is dispersed among multiple cloud storage devices [24]. Projects are fragmented into critical tasks, so that access to several tasks cannot reveal substantial information about the entire project. To this end, at most $n(P)$ tasks of each project P can be stored on a single storage device. Viewing a cloud storage device as a *bin* and each project as a *group* containing a collection of critical tasks (*items*), the problem of storing a set of projects on a minimal number of (identical) storage devices yields an instance of BPP.

BPP arises also in machine scheduling. Consider the following variant of *equitable scheduling* on a single machine [25]. The input is a set of n clients, each having a collection of jobs with arbitrary processing times, and a single machine that is available at any day for a limited amount of time. To ensure fairness, at most k jobs of the same client can be processed in a single day, for some $k \geq 1$. The objective is to complete processing all the jobs in a minimum number of days, subject to the equitability constraints. An instance of the problem can be cast as an instance of BPP, where days are the bins, jobs are the items, and the set of jobs of each client forms a group.

Another application of BPP comes from clustering problems with fairness constraints, where we have a set of items, each belongs to certain *category*, and we seek a partition of the items into clusters (or, groups) with restriction on the number of items from each category selected to each cluster, to ensure fairness. One real-life example is grouping students for educational activities, in which there is some cost associated with each participating student, and an overall budget for each group. The goal is to partition the students into a minimal number of working groups subject to the budget constraints, such that each group is balanced in terms of protected attributes like gender or race (since studies indicate that students might learn better in a diverse group). This variant of the *multi-fair capacitated grouping problem* [39] yields an instance of BPP.

Let $\mathrm{OPT} = \mathrm{OPT}(\mathcal{I})$ be the value of an optimal solution for an instance $\mathcal{I}$ of a minimization problem $\mathcal{P}$. As in the Bin Packing problem, we distinguish between *absolute* and *asymptotic* approximation. For $\alpha \geq 1$, we say that $\mathcal{A}$ is an absolute α-approximation algorithm for $\mathcal{P}$ if for any instance $\mathcal{I}$ of $\mathcal{P}$ it holds that $\mathcal{A}(\mathcal{I})/\mathrm{OPT}(\mathcal{I}) \leq \alpha$, where $\mathcal{A}(\mathcal{I})$ is the value of the solution returned by $\mathcal{A}$. Algorithm $\mathcal{A}$ is an *asymptotic* α-approximation algorithm if $\mathcal{A}(\mathcal{I}) \leq \alpha\mathrm{OPT}(\mathcal{I}) + o(\mathrm{OPT}(\mathcal{I}))$ for any instance $\mathcal{I}$ of $\mathcal{P}$. An *asymptotic polynomial-time approximation scheme (APTAS)* is a family of algorithms $(\mathcal{A}_\varepsilon)_{\varepsilon>0}$ such that, for every $\varepsilon > 0$, $\mathcal{A}_\varepsilon$ is a polynomial-time asymptotic $(1+\varepsilon)$-approximation algorithm for $\mathcal{P}$. An *asymptotic* fully *polynomial-time approximation scheme (AFPTAS)* is an APTAS $(\mathcal{A}_\varepsilon)_{\varepsilon>0}$ such that $\mathcal{A}_\varepsilon(\mathcal{I})$ runs in time $\mathrm{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$, where $|\mathcal{I}|$ is the encoding length of the instance $\mathcal{I}$; that is, there is a bivariate polynomial p such that $\mathcal{A}_\varepsilon(\mathcal{I})$ runs in time $p(|\mathcal{I}|, \frac{1}{\varepsilon})$ or less.

[1] For any $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, 2, \ldots, n\}$.

By known results for the special case of Bin Packing [20], BP cannot be approximated within (absolute) ratio better than $\frac{3}{2}$, unless P=NP. Thus, we focus in this paper on deriving better *asymptotic* approximation ratio for BPP. Given that classic BP admits an AFPTAS [32, 26], it is natural to ask whether the same holds for the problem with the added partition matroid constraint. We answer this question affirmatively.

▶ **Theorem 1.** *There is an* AFPTAS *for* BPP.

As a special case, our scheme improves upon the recent APTAS of Doron-Arad et al. [8] for *group bin packing (GBP)*, where the cardinality constraint of all groups is equal to one (i.e., $k(G) = 1\ \forall G \in \mathcal{G}$). This problem has been studied since the mid-1990's [37, 31].[2] Theorem 1 also generalizes the AFTPASs of [15, 30] for *bin packing with cardinality constraints (BPCC)*, the special case of BPP where all items belong to a single group (i.e., $\mathcal{G} = \{I\}$).[3]

1.1 Our Technique

We derive our scheme via a new technique for rounding a (fractional) *configuration* LP (c-LP) solution. Our technique interprets the standard c-LP solution as a prototype for a solution which is then modified via a sequence of rounding steps; a polytope associated with the prototype is used to ensure the prototype represents a valid solution following each of the rounding steps.

High Level Approach

To obtain a packing of the input BPP instance $\mathcal{I}$, our scheme partitions the set of items I into subsets $I_1, \ldots, I_r$; for each subset of items I_j it generates a packing $A'_1, \ldots, A'_m$ of the *large* items $L_j \subseteq I_j$ (of sizes at least δ), and extends the packing (in the *packing phase*) to include the *small* items $S_j = I_j \setminus L_j$ using a greedy algorithm (see Section 2). For this procedure to work, the following crucial properties must be satisfied.

1. The size $s(\ell) \leq \delta$ of a small item $\ell \in S_j$ is much smaller than the free space $\delta \ll 1 - s(A'_b)$ in a bin A'_b.
2. There is an upper bound $k_{j,G}$ on the number of large items from each group $G \in \mathcal{G}$ in any bin A'_b.
3. For each group $G \in \mathcal{G}$, there cannot be "too many" small items from G in S_j.

Figure 1 illustrates the packing of items in I_j. By the above properties, a main part of our scheme deals with generation of a partition of I into $I_1, \ldots, I_r$ and the corresponding partial packings.[4]

Let $\mathcal{C}$ denote the set of configurations of a BPP instance $\mathcal{I} = (I, \mathcal{G}, s, k)$. Our rounding technique interprets the vectors $\bar{x} \in \mathbb{R}^{\mathcal{C}}_{\geq 0}$ (i.e., vectors having a non-negative entry for each configuration $C \in \mathcal{C}$) as *prototypes*. The prototype $\bar{x} \in \mathbb{R}^{\mathcal{C}}_{\geq 0}$ serves as a blueprint for a (fractional) packing. Within the context of prototypes, each item $\ell \in C$ of a configuration C is interpreted as a placeholder (or, "slot") for items which can replace it. Also, some items may be added, thus utilizing the available capacity of the configuration C (given by $1 - s(C)$), while the items replacing the slots utilize the capacity $s(C)$ of the configuration. The value $\bar{x}_C$ represents a solution in which $\bar{x}_C$ configurations match the blueprint corresponding to C.

[2] See Table 1 for a summary of known results for GBP.
[3] An outline of the organization of this paper is given in Section 1.3.
[4] See the details in Section 3.

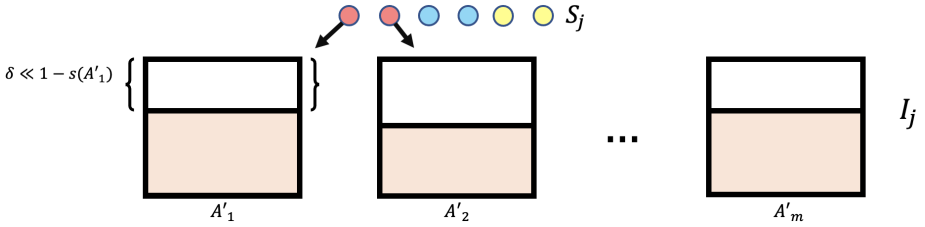

Figure 1 The set I_j in the partition $I_1, \ldots, I_r$, along with a partial packing $A'_1, \ldots, A'_m$ of the large items, L_j; the set S_j of remaining small items is represented by circles, whose colors indicate the groups. The items in S_j are added using a greedy algorithm.

A main building block in our scheme is an association of a polytope with each prototype. A non-empty polytope indicates the feasibility of a prototype. The partition of I into $I_1, \ldots, I_r$ and S, and the generation of the partial packings $(A'_1, \ldots, A'_m)$ relies on integrality properties of vertices in the polytope (associated with a prototype). However, for the integrality properties to be useful, we must obtain a *good prototype* $\bar{z}$, having a small number of configurations on the support (i.e, $\{C \in \mathcal{C} \mid \bar{z}_C > 0\}$), while the number of items in each configuration on the support must be small. We give below a high level description of how we construct a good prototype.

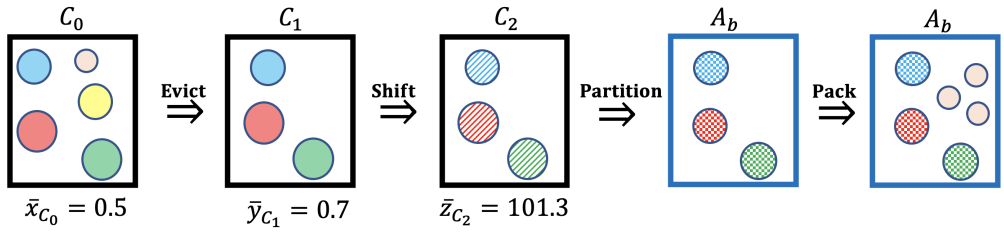

Figure 2 An illustration of the rounding process. Starting from a prototype $\bar{x}$, we follow a configuration C_0. The eviction process generates a prototype $\bar{y}$, where the value $\bar{x}_{C_0}$ is added to $\bar{y}_{C_1}$. In the shifting phase, $\bar{y}$ is transformed into a *good* prototype $\bar{z}$ in which items are replaced by a small number of representatives, and the entry $\bar{y}_{C_1}$ is added to $\bar{z}_{C_2}$. The partition phase uses the prototype $\bar{z}$ to define a partition $I_1, \ldots, I_r$ of the items, and an initial packing $A_1, \ldots, A_m$ for each I_j (containing the bin A_b in our example). Finally, in the packing phase, the remaining small items are added to the existing bins.

An initial prototype is obtained by solving a standard configuration-LP formulation of the problem. In this prototype, each item serves as a placeholder for itself. The algorithm modifies the prototype sequentially using two steps: *eviction* and *shifting*. The eviction step reduces the number of items per configuration on the support of the prototype. The shifting step reduces the overall number of distinct items used by configurations on the support. Thus, after the shifting, an item (or, a *slot*) may be used as a placeholder for many other items in the same group. Our construction in this step is non-trivial. Specifically, we use the *fractional grouping* technique introduced in [16] *constructively* (see the full version of the paper [9]). By the above, the number of configurations on the support of the constructed prototype is small, and our scheme can easily find a packing of the instance. We illustrate the main components of our multi-step rounding process in Figure 2.

Our technique may be useful in solving other packing problems, including, e.g., *bin packing with matching constraints* (see Section 4).

1.2 Prior Work

The special case of Group Bin Packing (GBP) (in which $k(G) = 1$ for all $G \in \mathcal{G}$) was first studied by Oh and Son [37]. This problem is a special case of *bin packing with conflicts (BPC)*, where the conflict graph is a *cluster graph* (see, e.g., [11] for a survey on recent results on BPC). An approximation ratio of 2.5 follows from the results of Jansen and Öhring [31] (for a generalization of GBP); this ratio was later improved by Epstein and Levin [14]. Better constants were given in several papers (e.g., [36, 1]). The best known asymptotic approximation for GBP prior to our work is an APTAS due to Doron-Arad et al. [8]. We summarize the known results for GBP in Table 1.

Table 1 Known Results for the special case of Group Bin Packing; $V(\mathcal{I}) = \max_{G \in \mathcal{G}} |G|$. The results of [31, 14, 1] for GBP follow as a special case from results for more general problems.

Authors	Year	Approximation
Oh and Son [37]	1995	$1.7 \cdot \mathrm{OPT}(\mathcal{I}) + 2.19 \cdot V(\mathcal{I})$
Jansen and Öhring [31]	1997	$2.5 \cdot \mathrm{OPT}(\mathcal{I})$
McCloskey and Shankar [36]	2005	$2 \cdot \mathrm{OPT}(\mathcal{I}) + V(\mathcal{I})$
Epstein and Levin [14]	2008	$\frac{7}{3} \cdot \mathrm{OPT}(\mathcal{I})$
Adany et al. [1]	2013	$2 \cdot \mathrm{OPT}(\mathcal{I})$
Doron-Arad et al. [8]	2021	APTAS
This paper	2023	**AFPTAS**

The APTAS of [8] for GBP is based on extensive guessing of properties of an optimal solution which are then used as guidance for the assignment of items to bins; such enumeration is commonly used in studies of BP (e.g., [3]). We note that the algorithm of [8] does not round a solution for a configuration-LP, and cannot be viewed as a rounding algorithm in general. The extensive guessing leads to running time that is exponential in $\frac{1}{\varepsilon}$. For a special case of GBP, where the maximum cardinality of a group is some constant, an AFPTAS follows from a result of [27].

GBP was studied also in the context of scheduling on identical machines. Das and Wiese [6] introduced the problem of *makespan minimization with bag constraints*. In this generalization of the classic makespan minimization problem, each job belongs to a *bag*. The goal is to schedule the jobs on a set of m identical machines, for some $m \geq 1$, such that no two jobs in the same bag are assigned to the same machine, and the makespan is minimized. Das and Wiese [6] developed a PTAS for the problem with bag constraints. Later, Grage et al. [21] obtained an EPTAS.

Another special case of BPP is Bin Packing with Cardinality Constraints (BPCC), in which $|\mathcal{G}| = 1$, i.e., we have a single group G with $k(G) = k$, for some integer $k \geq 1$. BPCC has been studied since the 1970's [34, 35, 33, 4]. Epstein and Levin [15] presented an AFPTAS which relies on rounding a non-standard configuration-LP formulation of the problem. Later, Jansen et al. [30] gave an AFPTAS with improved additive term. The AFPTASs of [15, 30] rely on the property that if $k < \frac{1}{\varepsilon}$ then only $\frac{1}{\varepsilon}$ items fit into a bin; thus, linear shifting can be applied to the whole instance. We note that in a BPP instance (which consists of multiple groups) the cardinality bound for each group may be small (or even equal to 1), yet the number of items that can be packed in a single bin may be arbitrarily large. This is one of several hurdles encountered when attempting to adapt the techniques of [15, 30] to our setting of BP with a partition matroid constraint.

In the *matroid partitioning* problem, we are given a ground set U and a matroid $\mathcal{M} = (U, \mathcal{S})$, where $\mathcal{S}$ is a family of subsets of U, known as the *independent sets* of the matroid. We seek a partition of U into as few independent sets as possible. The problem is polynomially solvable for any matroid $\mathcal{M}$ over U using a combinatorial algorithm (see, e.g., [12, 19]). When $\mathcal{M}$ is a partition matroid, the matroid partitioning problem can be viewed as a variant of BPP with unbounded bin capacities.

The use of configuration-LP (c-LP) in approximation algorithms started in the seminal paper of Karmakar and Karp on BP [32]. Their approach is to round the item sizes prior to solving a c-LP. Similar approaches, in which item sizes are rounded or the instance is restructured prior to solving the c-LP, can be found also in later works (e.g., [15]). Other works obtain an integral solution by applying randomized rounding to the solution of a standard c-LP. This includes the *Round & Approx* technique of Bansal et al. [2] and the tight approximation for the *separable assignment problem (SAP)* due to Fleischer et al. [18]. In [28] and [29], Jansen combines techniques for *2D strip packing* to round the solution of c-LP for multiple knapsack. Our deterministic rounding technique deviates significantly from these known approaches.

To the best of our knowledge, Bin Packing with Partition Matroid is studied here for the first time.

1.3 Organization

In Section 2 we include some definitions and notation. Section 3 gives an overview of our scheme, that is applied to a *structured* instance. Due to space constraints, the technical sections along with most of the proofs are relegated to the full version of the paper [9]. In Section 4 we give a summary and directions for future work.

2 Preliminaries

Let $\mathcal{A}$ be an algorithm that accepts as input $\varepsilon > 0$. We say the running time of $\mathcal{A}$ is $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ if there is a two-variable polynomial $p(x, y)$ such that $p(|\mathcal{I}|, \frac{1}{\varepsilon})$ is an upper bound on the running time of $\mathcal{A}(\mathcal{I}, \varepsilon)$. To allow a simpler presentation of the results, we assume throughout the paper that the set of items I is $\{1, 2, \ldots, n\}$, and the items are sorted in non-increasing order by sizes, i.e., $s(1) \geq s(2) \geq \ldots s(n)$.

2.1 Tackling the Small Items

The classic asymptotic approximation schemes for Bin Packing (see, e.g., [17, 32]) rely on the key property that *small* items can be added to a partial packing of the instance with little overhead, using simple algorithms such as First-Fit (see, e.g., [40]). We note that packing small items in the presence of a partition matroid constraint is more involved. Even for the special case of BP with a cardinality constraint, the small items cannot be packed using simple classic BP heuristics (see, e.g., [15]).

We show below that an efficient packing of the small items of a BPP instance can still be found using a relatively simple algorithm; however, the setting in which the algorithm can be applied is more restrictive, and items cannot be easily added to a partial packing of the instance (i.e., a set of configurations). Furthermore, the quality of such a packing depends on the *cardinality bound* of the BPP instance $\mathcal{I} = (I, \mathcal{G}, s, k)$, defined by $V(\mathcal{I}) = \max_{G \in \mathcal{G}} \left\lceil \frac{|G|}{|k(G)|} \right\rceil$. Formally,

▶ **Lemma 2.** *Given a* BPP *instance* $\mathcal{I} = (I, \mathcal{G}, s, k)$ *and* $\delta \in (0, 0.5)$*, such that* $s(\ell) \leq \delta$ *for all* $\ell \in I$*, there is an algorithm* Greedy *that returns in polynomial time a packing of* $\mathcal{I}$ *in at most* $(1 + 2\delta) \cdot \max\{s(I),\ V(\mathcal{I})\} + 2$ *bins.*

The proof of Lemma 2 is given in [9].

2.2 Structuring the Instance

Our scheme initially transforms a given BPP instance into one having a structure which depends on the parameter $\varepsilon > 0$. In this new instance, only a small number of groups may contain relatively large items. Let $K : (0, 0.1) \to \mathbb{R}$, where $K(\varepsilon) = \varepsilon^{-\varepsilon^{-2}}$ for all $\varepsilon \in (0, 0.1)$. We use K for defining a structured instance.

▶ **Definition 3.** *Given a BPP instance* $\mathcal{I} = (I, \mathcal{G}, s, k)$ *and* $\varepsilon \in (0, 0.1)$*, we say that* $\mathcal{I}$ *is* ε-structured *if there is* $\mathcal{B} \subseteq \mathcal{G}$ *such that* $|\mathcal{B}| \leq K(\varepsilon)$ *and for all* $G \in \mathcal{G} \setminus \mathcal{B}$ *and* $\ell \in G$ *it holds that* $s(\ell) < \varepsilon^2$*.*

Following the structuring step, our scheme proceeds to solve BPP on the structured instance. As a final step, the packing found for the structured instance is transformed into a packing of the original instance. This is formalized in the next result.

▶ **Lemma 4.** *There is a pair of algorithms,* Reduce *and* Reconstruct*, which satisfy the following.*

1. *Given a* BPP *instance* $\mathcal{J}$ *and* $\varepsilon > 0$ *such that* $\varepsilon^{-1} \in \mathbb{N}$*, algorithm* Reduce *returns in time* $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *an* ε*-structured BPP instance* $\mathcal{I}$*, where* $\text{OPT}(\mathcal{I}) \leq \text{OPT}(\mathcal{J})$*.*
2. *Given a* BPP *instance* $\mathcal{J}$*,* $\varepsilon > 0$ *such that* $\varepsilon^{-1} \in \mathbb{N}$*, and a packing* A' *for* $\mathcal{I} = \text{Reduce}(\mathcal{J}, \varepsilon)$ *of size* m'*, algorithm* Reconstruct *returns in time* $\text{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a packing* A *for the instance* $\mathcal{J}$ *of size* m*, where* $m \leq m' + 13\varepsilon \cdot \text{OPT}(\mathcal{J}) + 1$*.*

The structured instance $\mathcal{I}$ is obtained from $\mathcal{J}$ by reassigning items of size at least ε^2 from all but a few groups to a new group. The reconstruction algorithm modifies the packing of $\mathcal{I}$ such that each bin in the solution is a configuration of $\mathcal{J}$. The proof of Lemma 4 (given in [9]) is inspired by ideas of [6, 21, 8]. By Lemma 4, an AFTPAS for ε-structured BPP instances implies an AFTPAS for general BPP instances.

3 Approximation Algorithm for ε-Structured Instances

In this section we give an overview of our scheme. For $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, let $\mathcal{I} = (I, \mathcal{G}, s, k)$ be an ε-structured BPP instance. Recall that a *configuration* of $\mathcal{I}$ is a subset of items $C \subseteq I$ such that $\sum_{\ell \in C} s(\ell) \leq 1$, and $|C \cap G| \leq k(G)$ for all $G \in \mathcal{G}$. Let $\mathcal{C}(\mathcal{I})$ be the set of all configurations of $\mathcal{I}$; we use $\mathcal{C}$ when the instance $\mathcal{I}$ is clear from the context. Also, for every item $\ell \in I$ let $\mathcal{C}[\ell] = \{C \in \mathcal{C} \mid \ell \in C\}$ be the set of configurations of $\mathcal{I}$ that contain ℓ. A key component in our scheme is the construction of a *prototype* of a packing; a prototype gives a non-negative value to each configuration, which (informally) indicates the selection of the configuration. Specifically,

▶ **Definition 5.** *Given a* BPP *instance* $\mathcal{I}$*, a* prototype *is a vector* $\bar{x} \in \mathbb{R}^{\mathcal{C}}_{\geq 0}$*.*

In the context of prototypes, each configuration $C \in \mathcal{C}$ is interpreted as a set of placeholders called *types*. Each item $j \in C$ is a *slot-type*, which is a placeholder for a smaller or equally sized item of the same group. Also, the unused capacity of the configuration C (i.e., $1 - s(C)$) serves

as a placeholder for additional items; we refer to this placeholder as the *configuration-type* C. Thus, C is interpreted as the set of slot-types of j for each $j \in C$ and the configuration-type of C itself.

Intuitively, a slot-type (configuration-type) can be replaced by an item (items) which *fit* into it. For any $j \in I$ define $\mathsf{group}(j) = G$, where $G \in \mathcal{G}$ is the unique group such that $j \in G$. The subset of items that fit in place of the slot-type $j \in I$ is

$$\mathsf{fit}(j) = \{\ell \in \mathsf{group}(j) \mid s(\ell) \leq s(j)\} \qquad \forall j \in I. \tag{1}$$

For our algorithm to work, the items which fit into the configuration-type $C \in \mathcal{C}$ must be *small* relative to the unused capacity of C. We define the subset of items that fit into the configuration-type $C \in \mathcal{C}$ by

$$\mathsf{fit}(C) = \{\ell \in I \mid s(\ell) \leq \min\{\varepsilon^2, \varepsilon \cdot (1 - s(C))\}\} \qquad \forall C \in \mathcal{C}. \tag{2}$$

In words, $\mathsf{fit}(C)$ contains items of sizes smaller than ε^2 and also at most ε-fraction of the unused capacity of C. Figure 3 illustrates the above definitions.

For any prototype $\bar{x}$ we define the *$\bar{x}$-polytope* as a set of fractional packings, in which items are fractionally assigned to slot-types and configuration-types. The set of *types* of the instance $\mathcal{I}$ is $I \cup \mathcal{C}$. That is, the slot-types I and configuration-types $\mathcal{C}$. A point in the $\bar{x}$-polytope has an entry for each pair of an item $\ell \in I$ and a type $t \in I \cup \mathcal{C}$ which represents the fractional *assignment* of the item to the type. Formally,

▶ **Definition 6.** *Given a* BPP *instance $\mathcal{I}$, the set $\mathcal{C}$ of configurations for $\mathcal{I}$, and a prototype $\bar{x}$ of $\mathcal{I}$, the $\bar{x}$-polytope is the set containing all points $\bar{\gamma} \in [0,1]^{I \times (I \cup \mathcal{C})}$ which satisfy the following constraints.*

$$\bar{\gamma}_{\ell,t} = 0 \qquad \forall \ell \in I, t \in I \cup \mathcal{C} \text{ s.t. } \ell \notin \mathsf{fit}(t) \tag{3}$$

$$\sum_{\ell \in I} \bar{\gamma}_{\ell,C} \cdot s(\ell) \leq (1 - s(C)) \cdot \bar{x}_C \qquad \forall C \in \mathcal{C} \tag{4}$$

$$\sum_{\ell \in G} \bar{\gamma}_{\ell,C} \leq \bar{x}_C \cdot (k(G) - |C \cap G|) \qquad \forall G \in \mathcal{G}, C \in \mathcal{C} \tag{5}$$

$$\sum_{\ell \in I} \bar{\gamma}_{\ell,j} \leq \sum_{C \in \mathcal{C}[j]} \bar{x}_C \qquad \forall j \in I \tag{6}$$

$$\sum_{t \in I \cup \mathcal{C}} \bar{\gamma}_{\ell,t} \geq 1 \qquad \forall \ell \in I \tag{7}$$

Constraints (3) indicate that an item $\ell \in I$ cannot be assigned to type t if ℓ does not fit in t. Constraints (4) set an upper bound on the total (fractional) size of items assigned to each configuration-type $C \in \mathcal{C}$. This bound is equal to the residual capacity of C times the fractional number of bins packed with C, given by $\bar{x}_C$. Constraints (5) bound the number of items in each group G assigned to configuration-type C; at most $k(G) - |C \cap G|$ items in G can be added to C without violating the cardinality constraint of G. Constraints (6) bound the number of items assigned to slot-type $j \in I$ by the total selection of configurations containing j in $\bar{x}$. Finally, constraints (7) guarantee that each item is fully assigned to the types.

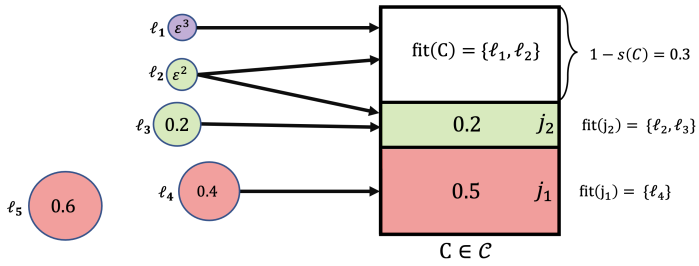

Figure 3 An example of items $\ell_1, \ldots, \ell_5$ of sizes $\varepsilon^3, \varepsilon^2, 0.2, 0.4, 0.6$, respectively, along with a configuration $C = \{j_1, j_2\} \in \mathcal{C}$ interpreted as the slot-types j_1, j_2 of sizes $0.5, 0.2$ and the configuration-type C. The colors of the items and slot-types indicate their corresponding groups. An arrow indicates that the item on the left fits with a slot-type or with the configuration-type C.

Let $\mathsf{supp}(\bar{x}) = \{C \in \mathcal{C} \mid \bar{x}_C > 0\}$ be the *support* of $\bar{x}$. Throughout this paper, we use prototypes $\bar{x}$ for which $\mathsf{supp}(\bar{x})$ is polynomial in the input size; thus, these prototypes have sparse representations. Our scheme first converts an initial prototype $\bar{x}$ (defined by a solution for the configuration-LP) into a *good* prototype $\bar{z}$ as defined below, and then constructs a packing based on $\bar{z}$. Let $Q : (0, 0.1) \to \mathbb{R}$ where $Q(\varepsilon) = \exp(\varepsilon^{-17})$ for all $\varepsilon \in (0, 0.1)$. Then,

▶ **Definition 7.** *Given $\varepsilon \in (0, 0.1)$ and an ε-structured* BPP *instance $\mathcal{I}$, a prototype $\bar{z}$ of $\mathcal{I}$ is a* good prototype *if the $\bar{z}$-polytope is non-empty, $|\mathsf{supp}(\bar{z})| \leq Q(\varepsilon)$, and $|C| \leq \varepsilon^{-10}$ for all $C \in \mathsf{supp}(\bar{z})$.*

To construct a good prototype, our algorithm first finds an initial prototype $\bar{x}$ using a Configuration-LP of the problem.[5] However, $\bar{x}$ is not necessarily a good prototype, since it may have a support of large size. Therefore, we apply a non-trivial rounding process, starting from the initial prototype $\bar{x}$, and eventually generate a good prototype $\bar{z}$. We consider this rounding process, along with the notions of prototype and $\bar{x}$-polytope, the core technical contribution of this paper. The next result summarizes the main properties of our algorithm for finding a good prototype, presented in Section 3.1. We use $\|\bar{x}\| = \sum_{C \in \mathcal{C}} \bar{x}_C$ to denote the ℓ_1-norm of a prototype $\bar{x}$.

▶ **Lemma 8.** *There is an algorithm* Prototype *that given $\varepsilon \in (0, 0.1)$ and an ε-structured* BPP *instance $\mathcal{I}$, returns in time* $\mathrm{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a good prototype $\bar{z}$ of $\mathcal{I}$ such that $\|\bar{z}\| \leq (1 + 19\varepsilon) \cdot \mathrm{OPT}(\mathcal{I}) + Q(\varepsilon)$.*

Given a good prototype $\bar{z}$, our scheme finds an efficient packing of the instance. This phase relies on integrality properties of the $\bar{z}$-polytope, combined with a matching-based algorithm and a greedy assignment of relatively small items using Algorithm Greedy (Lemma 2).

▶ **Lemma 9.** *There is an algorithm* Solution *that given $\varepsilon \in (0, 0.1)$, an ε-structured* BPP *instance $\mathcal{I}$, and a good prototype $\bar{z}$ of $\mathcal{I}$, returns in time* $\mathrm{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a packing of $\mathcal{I}$ in at most $(1 + 2\varepsilon) \cdot \|\bar{z}\| + 5\varepsilon^{-22} \cdot Q^2(\varepsilon)$ bins.*

The proof of Lemma 9 is given in Section 3.2. Using the above components, we obtain an AFPTAS for ε-structured instances. The pseudocode of the scheme is given in Algorithm 1.

▶ **Lemma 10.** *Given $\varepsilon \in (0, 0.1)$, $\varepsilon^{-1} \in \mathbb{N}$, and an ε-structured* BPP *instance $\mathcal{I}$, Algorithm 1 returns in time* $\mathrm{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a packing of $\mathcal{I}$ of size at most $(1 + 60\varepsilon) \cdot \mathrm{OPT}(\mathcal{I}) + (Q(\varepsilon))^3$.*

[5] See Section 3.1.

Algorithm 1 AFPTAS$(\mathcal{I}, \varepsilon)$.

Input : An ε-structured instance $\mathcal{I}$ and $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$
Output : A packing of $\mathcal{I}$
1 Find a good prototype $\bar{z} = \mathsf{Prototype}(\mathcal{I}, \varepsilon)$.
2 Return a packing $\Phi = \mathsf{Solution}(\mathcal{I}, \varepsilon, \bar{z})$.

Lemma 10 follows from Lemmas 8 and 9. Theorem 1 can be easily derived using Lemmas 10 and 4. We give the full proofs in [9].

3.1 Algorithm Prototype

In this section we present Algorithm Prototype which finds a good prototype for a given ε-structured instance. The algorithm uses an LP relaxation of the given BPP instance. For $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, let $\mathcal{I} = (I, \mathcal{G}, s, k)$ be an ε-structured BPP instance. Define the *configuration-LP* of $\mathcal{I}$ as:

$$\begin{aligned} \min \quad & \sum_{C \in \mathcal{C}} \bar{x}_C & & \\ \text{s.t.} \quad & \sum_{C \in \mathcal{C}[\ell]} \bar{x}_C = 1 & & \forall \ell \in I \\ & \bar{x}_C \geq 0 & & \forall C \in \mathcal{C} \end{aligned} \tag{8}$$

A solution for the LP (8) assigns to each configuration $C \in \mathcal{C}$ a real number $\bar{x}_C \in [0, 1]$ which indicates the fractional selection of C for the solution such that each item is fully *covered*. Observe that a solution for (8) is in particular a prototype of $\mathcal{I}$.

Note that the configuration-LP (8) has an exponential number of variables; thus, it cannot be solved in polynomial time by applying standard techniques. A common approach for solving such linear programs is to use a *separation oracle* for the dual program.

Consider the *configuration maximization problem (CMP)* in which we are given a BPP instance $\mathcal{I} = (I, \mathcal{G}, s, k)$ and a weight function $w : I \to \mathbb{R}_{\geq 0}$; the objective is to find a configuration $C \in \mathcal{C}$ such that $\sum_{\ell \in C} w(\ell)$ is maximized. By a well known connection between separation and optimization, an FPTAS for CMP implies an FPTAS for the configuration-LP of $\mathcal{I}$ [22, 18, 23, 38]. CMP can be solved via an easy reduction to *knapsack with partition matroid*, that is known to admit an FPTAS [10]. Thus, we have

▶ **Lemma 11.** *There is an algorithm* SolveLP *that given a* BPP *instance* $\mathcal{I}$ *and* $\varepsilon > 0$*, returns in time* $\mathrm{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a solution for the configuration-*LP *of* $\mathcal{I}$ *of value at most* $(1 + \varepsilon)\mathrm{OPT}$*, where* OPT *is the value of an optimal solution for the configuration-*LP *of* $\mathcal{I}$*.*

We give the proof of Lemma 11 in [9]. A solution $\bar{x}$ for the Configuration-LP (8) is a prototype of the instance such that the $\bar{x}$-polytope is non-empty; in particular, it contains the point $\bar{\gamma}$ where $\bar{\gamma}_{\ell,j} = 1$ $\forall \ell, j \in I$ such that $\ell = j$, and $\bar{\gamma}_{\ell,t} = 0$ otherwise (note that this property does not hold for the good prototype $\bar{z}$ returned by Algorithm Prototype). Our intermediate goal is an *evicted prototype* $\bar{y}$, having configurations of bounded cardinality on its support, but with properties similar to those of solutions for (8). We say that an item $\ell \in I$ is ε-large if $s(\ell) \geq \varepsilon^2$. We use $L(\varepsilon, \mathcal{I})$ to denote the set of ε-large items of an instance $\mathcal{I}$. If $\mathcal{I}$ and ε are known by context we simply use L (instead of $L(\varepsilon, \mathcal{I})$). We now formalize the above.

▶ **Definition 12.** *Let $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$ and $\mathcal{I}$ an ε-structured BPP instance. A prototype $\bar{y}$ of $\mathcal{I}$ is called an* evicted prototype *if the following holds.*

1. *For all $C \in \mathsf{supp}(\bar{y})$ it holds that $|C| \leq \varepsilon^{-10}$ and $s(C \setminus L) \leq \varepsilon$.*
2. *There exists $\bar{\gamma}$ in the $\bar{y}$-polytope such that $\bar{\gamma}_{\ell,j} = 0$ for all $\ell, j \in I$ where $\ell \neq j$.*
3. *$\sum_{C \in \mathcal{C}[\ell]} \bar{y}_C \leq 2$ for every $\ell \in I$.*

Given a solution $\bar{x}$ for the configuration-LP (8), we construct an evicted prototype $\bar{y}$ with $\|\bar{y}\| \approx \|\bar{x}\|$. Our technique fractionally maps each configuration $C \in \mathsf{supp}(\bar{x})$ to other configurations, where relatively small items are discarded in the mapping. To show that the $\bar{y}$-polytope is non-empty, we generate a point in the $\bar{y}$-polytope by assigning (fractionally) discarded items to configuration-types (see Definition 6). The result of this process is outlined in the next lemma.

▶ **Lemma 13.** *There is an algorithm* Evict *which given $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, an ε-structured* BPP *instance $\mathcal{I}$, and a solution $\bar{x}$ for the configuration-LP* (8)*, returns in time* $\mathrm{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *an evicted prototype $\bar{y}$ such that $\|\bar{y}\| \leq (1 + \varepsilon)\|\bar{x}\|$.*

A complete presentation of algorithm Evict and the proof of Lemma 13 are given in [9]. Observe that property 2 of Definition 12 allows $\bar{\gamma}_{\ell,C} > 0$ for item $\ell \in I$ and a configuration-type $C \in \mathcal{C}$; the property that $\bar{\gamma}_{\ell,j} > 0$ for $\ell \neq j$ is obtained only in the next step. Moreover, note that Evict does not return the vector $\bar{\gamma}$ but only guarantees its existence.

Given an evicted prototype $\bar{y}$, our scheme uses algorithm Shift to generate a good prototype $\bar{z}$ with $\|\bar{z}\| \approx \|\bar{y}\|$. As in algorithm Evict, we rely on a fractional mapping between configurations to construct $\bar{z}$; here, our goal is to significantly decrease $|\mathsf{supp}(\bar{z})|$ w.r.t. $|\mathsf{supp}(\bar{y})|$ while keeping the $\bar{z}$-polytope non-empty. One key observation utilizes combinatorial properties of the instance to show that items from most groups can be discarded in the mapping (the items may be assigned to configuration-types in the $\bar{z}$-polytope). Items of the remaining groups are mapped to a small number of *representatives*, using a non-trivial application of fractional grouping.

▶ **Lemma 14.** *There is an Algorithm* Shift *that given $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, an ε-structured* BPP *instance $\mathcal{I}$ and an evicted prototype $\bar{y}$, returns in time* $\mathrm{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a good prototype $\bar{z}$ such that $\|\bar{z}\| \leq (1 + 5\varepsilon)\|\bar{y}\| + Q(\varepsilon)$.*

An elaborate presentation of Algorithm Shift and the proof of Lemma 14 are given in [9]. Finally, Algorithm Prototype finds a good prototype by computing Algorithms SolveLP, Evict, and Shift sequentially. We give the pseudocode in Algorithm 2. The proof of Lemma 8 easily follows from Lemmas 11, 13, and 14; we give the proof in [9].

Algorithm 2 Prototype$(\mathcal{I}, \varepsilon)$.

Input : An ε-structured instance $\mathcal{I}$ and $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$
Output : A good prototype

1 Find a solution for the configuration-LP of $\mathcal{I}$; that is, $\bar{x} = \mathsf{SolveLP}(\mathcal{I}, \varepsilon)$.
2 Find an evicted prototype $\bar{y} = \mathsf{Evict}(\mathcal{I}, \bar{x}, \varepsilon)$.
3 Return a good prototype $\bar{z} = \mathsf{Shift}(\mathcal{I}, \bar{y}, \varepsilon)$.

3.2 Algorithm Solution

In this section we show how integrality properties of a good prototype yield an efficient packing of the instance. For $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, let $\mathcal{I} = (I, \mathcal{G}, s, k)$ be an ε-structured BPP instance. The next lemma shows that if a prototype $\bar{x}$ has a small support, and each configuration in the support contains a few items (as for good prototypes), then the vertices of the $\bar{x}$-polytope are almost integral. Thus, given a vertex $\bar{\lambda}$ of such $\bar{x}$-polytope, the items assigned fractionally by $\bar{\lambda}$ can be packed using only a small number of extra bins.

▶ **Lemma 15.** *Let $\mathcal{I}$ be a* BPP *instance, $k \in \mathbb{N}_{\geq 1}$, and a prototype $\bar{x}$ of $\mathcal{I}$ such that for all $C \in \mathsf{supp}(\bar{x})$ it holds that $|C| \leq k$ and $\bar{x}_C \in \mathbb{N}$. Then, for any vertex $\bar{\lambda}$ in the $\bar{x}$-polytope for which constraints* (7) *hold with equality,*

$$\left|\left\{\ell \in I \mid \exists t \in I \cup \mathcal{C} \text{ s.t. } \bar{\lambda}_{\ell,t} \in (0,1)\right\}\right| \leq 8k^2 \cdot |\mathsf{supp}(\bar{x})|^2.$$

The proof of Lemma 15 bears some similarity to a proof of [7], which shows the integrality properties of a somewhat different polytope. We give the detailed proof in [9].

Given a good prototype $\bar{z}$, our scheme finds a packing of the instance using a partition of the items into slot-types and configuration-types; intuitively, items assigned to slot-types are already packed (using the integrality property of the $\bar{z}$-polytope) and items assigned to configuration-types will be added using Greedy. This relies on the following construction.

For configurations $S, C \in \mathcal{C}$, we say that S is *allowed* in C if each item $\ell \in S$ can be mapped to a distinct slot $j \in C$, such that $\ell \in \mathsf{fit}(j)$. We consider packings of a subset of items to which we call a *category*. Each category is associated with (i) a configuration $C \in \mathcal{C}$ such that all bins in the category are allowed in C, and (ii) a *completion*: a subset of (unpacked) items bounded by total size and number of items per group, where each item fits with C. This is formalized in the next definition.

▶ **Definition 16.** *Let $\varepsilon \in (0, 0.1)$, an ε-structured BPP instance $\mathcal{I} = (I, \mathcal{G}, s, k)$, a configuration $C \in \mathcal{C}$, a packing $B = (B_1, \ldots, B_m)$ of a subset of I (category), and $D \subseteq I$ (completion). We say that B is a category of C and D if the following holds.*

- *For any $i \in [m]$, B_i is allowed in C.*
- *$D \subseteq \mathsf{fit}(C)$.*
- *$s(D) \leq (1 - s(C)) \cdot m$.*
- *For any $G \in \mathcal{G}$ it holds that $|D \cap G| \leq m \cdot (k(G) - |C \cap G|)$.*

The motivation behind this construction, is that Algorithm Greedy (Lemma 2) can be used to assign the completion to the existing bins of the category using only a small number of extra bins. Thus, our end-goal from the good prototype $\bar{z}$ is to obtain an *ε-nice partition*: a packing of a subset of I such that the bins in the packing are partitioned into a bounded number of categories; also, we require that each item $\ell \in I$ is either in this packing or in a completion of a category. The above constraints are analogous to constraints (3)-(7) of the $\bar{z}$-polytope, that is used for finding an assignment of the items to slots and configurations. This is formalized in Definition 17. An example is given in Figure 4.

▶ **Definition 17.** *Given $\varepsilon \in (0, 0.1)$, an ε-structured BPP instance $\mathcal{I} = (I, \mathcal{G}, s, k)$, an* ε-nice partition *$\mathcal{B}$ of $\mathcal{I}$ is a packing $(A_1, ..., A_m)$ of a subset of I, configurations $\mathcal{H} \subseteq \mathcal{C}$, categories $(B_C)_{C \in \mathcal{H}}$, and completions $(D_C)_{C \in \mathcal{H}}$ such that the following holds.*

- *$|\mathcal{H}| \leq \varepsilon^{-22} Q^2(\varepsilon)$.*
- *$\{B_C\}_{C \in \mathcal{H}}$ is a partition of $\{A_i \mid i \in [m]\}$.*
- *$\{D_C\}_{C \in \mathcal{H}}$ is a partition of $I \setminus \bigcup_{i \in [m]} A_i$.*
- *For all $C \in \mathcal{H}$ it holds that B_C is a category of C and D_C.*

The size *of $\mathcal{B}$ is m.*

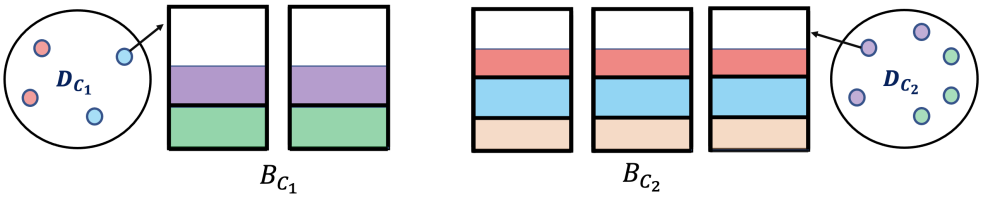

Figure 4 An example of an ε-nice partition, which consists of a packing in five bins partitioned into two categories $\mathcal{H} = \{B_{C_1}, B_{C_2}\}$. In addition, D_{C_1} and D_{C_2} are the completions of B_{C_1} and B_{C_2}, respectively. Colors indicate groups: if the cardinality bound of the blue group is 1, then D_{C_1} contains at most 2 blue items and D_{C_2} cannot contain blue items.

To obtain an ε-nice partition, Algorithm Partition initially rounds up the entries of $\bar{z}$ to obtain the prototype $\bar{z}^*$. It then finds a vertex $\bar{\lambda}$ of the $\bar{z}^*$-polytope, which is almost integral by Lemma 15. Thus, with the exception of a small number of items, each item is fully assigned either to a slot or to a configuration. Algorithm Partition uses $\bar{\lambda}$ to construct an ε-nice partition. We generate a category for each $C \in \mathsf{supp}(\bar{z}^*)$ and let $D_C = \{\ell \in I \mid \bar{\lambda}_{\ell,C} = 1\}$ be the set of all items assigned to C. We also generate $\bar{z}^*_C$ copies (bins) of each configuration and replace its slots by items via matching.

▶ **Lemma 18.** *There is an algorithm* Partition *that given $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, an ε-structured BPP instance $\mathcal{I}$, and a good prototype $\bar{z}$ of $\mathcal{I}$, returns in time* $\mathrm{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *an ε-nice partition of $\mathcal{I}$ of size at most $\|\bar{z}\| + \varepsilon^{-22}Q^2(\varepsilon)$.*

Algorithm Partition is presented in [9]. Given an ε-nice partition of size m, a packing of the instance in roughly m bins is obtained using the next lemma.

▶ **Lemma 19.** *There is a polynomial-time algorithm* Pack *which given $\varepsilon \in (0, 0.1)$ such that $\varepsilon^{-1} \in \mathbb{N}$, an ε-structured BPP instance $\mathcal{I}$, and ε-nice partition of $\mathcal{I}$ of size m, returns in time* $\mathrm{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ *a packing of $\mathcal{I}$ in at most $(1+2\varepsilon)m + 2\varepsilon^{-22}Q^2(\varepsilon)$ bins.*

Algorithm Pack utilizes Algorithm Greedy to add the items in a completion of a category to the bins of this category, possibly using a few extra bins. Algorithm Pack and Algorithm Greedy are presented in [9]. Finally, we construct Algorithm Solution, which first finds an ε-nice partition and then use it to find a packing for the instance. We give the pseudocode in Algorithm 3.

Proof of Lemma 9. By Lemma 18, $\mathcal{B}$ is an ε-nice partition with size at most $\|\bar{z}\| + \varepsilon^{-22} \cdot Q^2(\varepsilon)$. Then, by Lemma 19, Φ is a full packing of $\mathcal{I}$ using at most $(1+2\varepsilon) \cdot \|\bar{z}\| + 5\varepsilon^{-22} \cdot Q^2(\varepsilon)$ bins. The running time is $\mathrm{poly}(|\mathcal{I}|, \frac{1}{\varepsilon})$ by Lemmas 18, 19. ◀

Algorithm 3 Solution$(\mathcal{I}, \bar{z}, \varepsilon)$.

Input : $\varepsilon \in (0, 0.1)$, $\varepsilon^{-1} \in \mathbb{N}$, an ε-structured instance $\mathcal{I}$, a good prototype $\bar{z}$ of $\mathcal{I}$
Output : A packing of $\mathcal{I}$
1 Find an ε-nice partition $\mathcal{B}$ of $\mathcal{I}$ by Partition$(\mathcal{I}, \bar{z}, \varepsilon)$.
2 Return a packing $\Phi =$ Pack$(\mathcal{I}, \mathcal{B}, \varepsilon)$.

4 Discussion

In this paper we present an AFPTAS for Bin Packing with Partition Matroid. While BPP is a natural generalization of Bin Packing variants that have been studied in the past, to the best of our knowledge it is studied here for the first time. Our result improves upon the APTAS of [8] for the well studied special case of Group Bin Packing, and generalizes the AFPTASs of [15, 30] for the special case of Bin Packing with Cardinality Constraints. Our scheme applies a novel rounding method to solutions of the configuration-LP formulation of the problem. The rounding process relies on the key notion of a *prototype*, in which items are placeholders for other items, and sophisticated use of fractional grouping [16]. Our scheme demonstrates the power of this fractional version of linear grouping in solving constrained packing problems; it also shows how fractional grouping can be used *constructively*.

The rounding method introduced in this paper seems useful also for other settings. A preliminary study shows we can apply our method to obtain a polynomial time approximation scheme for *Multiple Knapsack with Partition Matroid*, a generalization of the Multiple Knapsack problem (see, e.g., [5, 28]) in which the items assigned to each bin form an independent set of a partition matroid. Furthermore, we can derive approximation algorithms for Machine Scheduling with Partition Matroid, a generalization of the classic Machine Scheduling problem in which the jobs assigned to a machine must be an independent set of a partition matroid. We note that this problem is a generalization of Machine Scheduling with Bag Constraints studied in [6, 21].

Another intriguing direction for future work is to apply our framework to Bin Packing with other types of constraints. We give two potential examples. The problem of Bin Packing with Partition Matroid is a special case of Bin Packing with Matroid, for which the input is a set of items I, a size function $s : I \to [0, 1]$ and a matroid $\mathcal{M}$. The objective is to partition I into a minimal number of bins $A_1, \ldots, A_m$ such that A_b is an independent set of the matroid $\mathcal{M}$, and $s(A_b) \leq 1$ for all $b \in [m]$. While this problem is a natural generalization of both Bin Packing and Matroid Partitioning, we were unable to find any published results. It would be interesting to obtain an APTAS for Bin Packing with Matroid using our framework. To this end, the reduction of the instance to a structured instance and the fractional shifting (see the full version of the paper [9]) need to be modified to tackle general matroids.

Finally, consider the Bin Packing with Matching Constraints problem (or, equivalently: Bin Packing with Line Graph Conflicts). The input is a graph $G = (V, E)$ and a size function on the edges $s : E \to [0, 1]$. The objective is to partition E into a minimal number of bins $A_1, \ldots, A_m$ such that $A_b \subseteq E$ is a matching in G, and $s(A_b) \leq 1$ for all $b \in [m]$. Our preliminary results suggest it may be possible to adapt the main components of our scheme for solving this problem.

References

1 Ron Adany, Moran Feldman, Elad Haramaty, Rohit Khandekar, Baruch Schieber, Roy Schwartz, Hadas Shachnai, and Tami Tamir. All-or-nothing generalized assignment with application to scheduling advertising campaigns. In *Integer Programming and Combinatorial Optimization - 16th International Conference, IPCO*, pages 13–24, 2013.

2 Nikhil Bansal, Alberto Caprara, and Maxim Sviridenko. A new approximation method for set covering problems, with applications to multidimensional bin packing. *SIAM Journal on Computing*, 39(4):1256–1278, 2010.

3 Nikhil Bansal, Marek Eliáš, and Arindam Khan. Improved approximation for vector bin packing. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on discrete algorithms*, pages 1561–1579. SIAM, 2016.

4 Alberto Caprara, Hans Kellerer, and Ulrich Pferschy. Approximation schemes for ordered vector packing problems. *Naval Research Logistics (NRL)*, 50(1):58–69, 2003.

5 Chandra Chekuri and Sanjeev Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 35(3):713–728, 2005.

6 Syamantak Das and Andreas Wiese. On minimizing the makespan when some jobs cannot be assigned on the same machine. In *25th Annual European Symposium on Algorithms, ESA*, pages 31:1–31:14, 2017.

7 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. An APTAS for bin packing with clique-graph conflicts. *arXiv preprint*, 2020. `arXiv:2011.04273`.

8 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. An APTAS for bin packing with clique-graph conflicts. In *17th International Symposium on Algorithms and Data Structures, WADS*, pages 286–299, 2021.

9 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. Bin packing with partition matroid can be approximated within $o(opt)$ bins. *arXiv preprint*, 2022. `arXiv:2212.01025`.

10 Ilan Doron-Arad, Ariel Kulik, and Hadas Shachnai. An FPTAS for Budgeted Laminar Matroid Independent Set. *arXiv preprint*, 2023. `arXiv:2304.13984`.

11 Ilan Doron-Arad and Hadas Shachnai. Approximating bin packing with conflict graphs via maximization techniques. *Proc. WG*, 2023.

12 Jack Edmonds. Minimum partition of a matroid into independent subsets. *J. Res. Nat. Bur. Standards Sect. B*, 69:67–72, 1965.

13 Ibtissam Ennajjar, Youness Tabii, and Abdelhamid Benkaddour. Securing data in cloud computing by classification. In *Proceedings of the 2nd international Conference on Big Data, Cloud and Applications*, pages 1–5, 2017.

14 Leah Epstein and Asaf Levin. On bin packing with conflicts. *SIAM Journal on Optimization*, 19(3):1270–1298, 2008.

15 Leah Epstein and Asaf Levin. AFPTAS results for common variants of bin packing: A new method for handling the small items. *SIAM Journal on Optimization*, 20(6):3121–3145, 2010.

16 Yaron Fairstein, Ariel Kulik, and Hadas Shachnai. Modular and submodular optimization with multiple knapsack constraints via fractional grouping. In *29th Annual European Symposium on Algorithms, ESA*, pages 41:1–41:16, 2021.

17 W Fernandez de La Vega and George S. Lueker. Bin packing can be solved within $1+\varepsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.

18 Lisa Fleischer, Michel X Goemans, Vahab S Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum separable assignment problems. *Mathematics of Operations Research*, 36(3):416–431, 2011.

19 Harold N Gabow and Herbert H Westermann. Forests, frames, and games: algorithms for matroid sums and applications. *Algorithmica*, 7(1):465–497, 1992.

20 Michael R Garey and David S Johnson. Computers and intractability. *A Guide to the*, 1979.

21 Kilian Grage, Klaus Jansen, and Kim-Manuel Klein. An EPTAS for machine scheduling with bag-constraints. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, pages 135–144, 2019.

22 Michael D Grigoriadis, Leonid G Khachiyan, Lorant Porkolab, and Jorge Villavicencio. Approximate max-min resource sharing for structured concave optimization. *SIAM Journal on Optimization*, 11(4):1081–1091, 2001.

23 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

24 Marcos Guerine, Murilo B Stockinger, Isabel Rosseti, Luidi G Simonetti, Kary ACS Ocaña, Alexandre Plastino, and Daniel de Oliveira. A provenance-based heuristic for preserving results confidentiality in cloud-based scientific workflows. *Future Generation Computer Systems*, 97:697–713, 2019.

25 Klaus Heeger, Danny Hermelin, George B Mertzios, Hendrik Molter, Rolf Niedermeier, and Dvir Shabtay. Equitable scheduling on a single machine. *Journal of Scheduling*, pages 1–17, 2022.

26 Rebecca Hoberg and Thomas Rothvoß. A logarithmic additive integrality gap for bin packing. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2616–2625. SIAM, 2017.

27 Klaus Jansen. An approximation scheme for bin packing with conflicts. *Journal of combinatorial optimization*, 3(4):363–377, 1999.

28 Klaus Jansen. Parameterized approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 39(4):1392–1412, 2010.

29 Klaus Jansen. A fast approximation scheme for the multiple knapsack problem. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 313–324. Springer, 2012.

30 Klaus Jansen, Marten Maack, and Malin Rau. Approximation schemes for machine scheduling with resource (in-) dependent processing times. *ACM Transactions on Algorithms (TALG)*, 15(3):1–28, 2019.

31 Klaus Jansen and Sabine R. Öhring. Approximation algorithms for time constrained scheduling. *Inf. Comput.*, 132(2):85–108, 1997.

32 Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science*, pages 312–320. IEEE, 1982.

33 Hans Kellerer and Ulrich Pferschy. Cardinality constrained bin-packing problems. *Annals of Operations Research*, 92:335–348, 1999.

34 Kenneth L Krause, Vincent Y Shen, and Herbert D Schwetman. Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. *Journal of the ACM (JACM)*, 22(4):522–550, 1975.

35 KL Krause, Vincent Y Shen, and Herbert D Schwetman. Errata:"analysis of several task-scheduling algorithms for a model of multiprogramming computer systems". *Journal of the ACM (JACM)*, 24(3):527, 1977.

36 Bill McCloskey and AJ. Shankar. *Approaches to bin packing with clique-graph conflicts*. Computer Science Division, University of California, 2005.

37 Y. Oh and S.H. Son. On a constrained bin-packing problem. *Technical Report CS-95-14*, 1995.

38 Serge A Plotkin, David B Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301, 1995.

39 Tai Le Quy, Gunnar Friege, and Eirini Ntoutsi. Multiple fairness and cardinality constraints for students-topics grouping problem. *arXiv preprint*, 2022. `arXiv:2206.09895`.

40 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.