All-or-Nothing Generalized Assignment with Application to Scheduling Advertising Campaigns

Ron Adany[1], Moran Feldman[2], Elad Haramaty[2], Rohit Khandekar[3], Baruch Schieber[4], Roy Schwartz[2], Hadas Shachnai[2], and Tami Tamir[5]

[1] Computer Science Department, Bar-Ilan University, Ramat-Gan 52900, Israel. adanyr@cs.biu.ac.il.

[2] Computer Science Department, Technion, Haifa 32000, Israel. E-mail: {moranfe,eladh,schwartz,hadas}@cs.technion.ac.il.

[3] Knight Capital Group, Jersey City, NJ 07310. E-mail: rkhandekar@gmail.com

[4] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598. E-mail: sbar@us.ibm.com

[5] School of Computer science, The Interdisciplinary Center, Herzliya, Israel. E-mail: tami@idc.ac.il.

Abstract. We study a variant of the *generalized assignment problem* (GAP) which we label *all-or-nothing* GAP *(*AGAP*)*. We are given a set of items, partitioned into n groups, and a set of m bins. Each item ℓ has size $s_\ell > 0$, and utility $a_{\ell j} \geq 0$ if packed in bin j. Each bin can accommodate at most one item from each group, and the total size of the items in a bin cannot exceed its capacity. A group of items is *satisfied* if all of its items are packed. The goal is to find a feasible packing of a subset of the items in the bins such that the total utility from satisfied groups is maximized. We motivate the study of AGAP by pointing out a central application in scheduling advertising campaigns.

Our main result is an $O(1)$-approximation algorithm for AGAP instances arising in practice, where each group consists of at most $m/2$ items. Our algorithm uses a novel reduction of AGAP to maximizing submodular function subject to matroid constraint. For AGAP instances with fixed number of bins, we develop a randomized *polynomial time approximation scheme (PTAS)*, relying on a non-trivial LP relaxation of the problem.

We present a $(3+\varepsilon)$-approximation as well as PTASs for other special cases of AGAP, where the utility of any item does not depend on the bin in which it is packed. Finally, we derive hardness results for the different variants of AGAP studied in the paper.

1 Introduction

Personalization of advertisements (ads) allows commercial entities to aim their ads at specific audiences, thus ensuring that each target audience receives its specialized content in the desired format. According to Nielsen [13, 14], $67B were spent on TV ads in the U.S. in 2010, and an average viewer watched TV for 158

hours per month, with this average consistently increasing. Based on these trends and on advances in cable TV technology, personalized TV ads are expected to increase revenues for TV media companies and for mobile operators [3, 8, 17]. The proliferation of alternative media screens, such as cell-phones and tablets, generate new venues for personalized campaigns targeted to specific viewers, based on their interests, affinity to the advertised content, and location. In fact, ads personalization is already extensively used on the Internet, e.g., in Google AdWords [6]. Our study is motivated by a central application in personalized ad campaigns scheduling, introduced to us by SintecMedia [19].

An *advertising campaign* is a series of advertisement messages that share a single idea and theme which make up an integrated marketing communication. Given a large set of campaigns that can be potentially delivered to the media audience, a service provider attempts to fully deliver a subset of campaigns that maximizes the total revenue, while satisfying constraints on the placement of ads that belong to the same campaign, as well as possible placement constraints among conflicting campaigns. In particular, to increase the number of viewers exposed to an ad campaign, one constraint is that each commercial break contains a single ad from this campaign.[6] Also, each ad has a given length (=size), which remains the same, regardless of the commercial break in which it is placed. This generic assignment problem defines a family of all-or-nothing variants of the *generalized assignment problem* (GAP).

Let $[k]$ denote $\{1, \ldots, k\}$ for an integer k. In *all-or-nothing* GAP (or AGAP), we are given a set of m bins, where bin $j \in [m]$ has capacity c_j, and a set of N items partitioned into n groups $G_1, \ldots, G_n$. Each group $i \in [n]$, consists of k_i items, for some $k_i \geq 1$, such that $\sum_i k_i = N$. Each item $\ell \in [N]$ has a size $s_\ell > 0$ and a non-negative utility $a_{\ell j}$ if packed in bin $j \in [m]$. An item can be packed in at most one bin, and each bin can accommodate at most one item from each group. Given a packing of a subset of items, we say that a group i is *satisfied* if all items in G_i are packed. The goal is to pack a subset of items in the bins so that the total utility of satisfied groups is maximized. Formally, we define a packing to be a function $p : [N] \to [m] \cup \{\perp\}$. If $p(\ell) = j \in [m]$ for $\ell \in [N]$, we say that item ℓ is packed in bin j. If $p(\ell) = \perp$, we say that item ℓ is not packed. A packing is *admissible* if $\sum_{\ell \in p^{-1}(j)} s_\ell \leq c_j$ for all $j \in [m]$ and $|p^{-1}(j) \cap G_i| \leq 1$ for all $j \in [m]$ and $i \in [n]$. Given a packing p, let $S_p = \left\{i \in [n] \mid G_i \subseteq \cup_{j \in [m]} \{p^{-1}(j)\}\right\}$ denote the set of groups satisfied by p. The goal in AGAP is to find an admissible packing p that maximizes the utility: $\sum_{i \in S_p} \sum_{\ell \in G_i} a_{\ell p(\ell)}$.

We note that AGAP is NP-hard already when the number of bins is fixed. Such instances capture campaign scheduling in a given time interval (of a few hours) during the day. We further consider the following special cases of AGAP, which are of practical interest. In *all-or-nothing group packing*, each group G_i has a profit $P_i > 0$ if all items are packed, and 0 otherwise. Thus, item utilities do not depend on the bins. In the *all-or-nothing assignment problem* (AAP), all items in G_i have the same size, $s_i > 0$, and same utility $a_i \geq 0$, across all bins.

[6] Indeed, overexposure of ads belonging to the same campaign in one break may cause lack of interest, thus harming the success of the campaign.

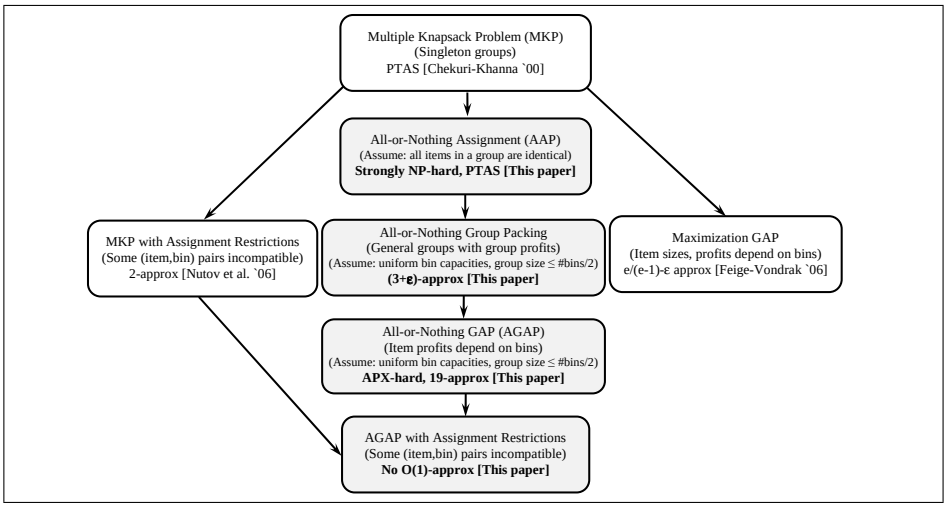

Fig. 1: Summary of our approximation and hardness results and comparison with related problems. An arrow from problem A to B indicates that A is a special case of B.

Note that the special case of AGAP where all groups consist of a *single* item yields an instance of classic GAP, where each item has the same size across the bins. The special case of AAP where all groups consist of a *single* item yields an instance of the multiple knapsack problem. Clearly, AGAP is harder to solve than these two problems. One reason is that, due to the *all-or-nothing* requirement, we cannot eliminate large items of small utilities, since these items may be essential for satisfying a set of most profitable groups. Moreover, even if the satisfied groups are known *a-priori*, since items of the same group cannot be placed in one bin, common techniques for classical packing, such as rounding and enumeration, cannot be applied.

1.1 Our Results

Figure 1 summarizes our contributions for different variants of AGAP and their relations to each other. Even relatively special instances of AAP are NP-hard. Furthermore, with slight extensions, AGAP becomes hard to approximate within any bounded ratio (see Appendix D). Thus, we focus in this paper on deriving approximation algorithms for AGAP and the above special cases.

Given an algorithm $\mathcal{A}$, let $\mathcal{A}(I), OPT(I)$ denote the utility of $\mathcal{A}$ and an optimal solution for a problem instance I, respectively. For $\rho \geq 1$, we say that $\mathcal{A}$ is a ρ-approximation algorithm if, for any instance I, $\frac{OPT(I)}{\mathcal{A}(I)} \leq \rho$.

We note that AGAP with non-identical bins is hard for any constant approximation, even if the utility of an item is identical across the bins (see Theorem 13). Thus, in deriving our results for AGAP, we assume the bins are of uniform capacities. Our main result (in Section 2) is a $(19 + \varepsilon)$-approximation algorithm

for AGAP instances arising in practice, where each group consists of at most $m/2$ items.

Interestingly, AGAP with a fixed number of bins admits a randomized PTAS (see Section 3.1). In Section 3.2 we show that, for the special case where all items have unit sizes, an $\frac{e}{e-1}$-approximation can be obtained by reduction to submodular maximization with knapsack constraint. In Section 3.3 we give a $(3+\varepsilon)$-approximation algorithm for All-or-Nothing Group Packing. This ratio can be improved to $(2+\varepsilon)$ if group sizes are relatively small.

In Appendix C we present PTASs for two subclasses of instances of AAP. The first is the subclass of instances with unit-sized items, the second is the subclass of instances in which item sizes are drawn from a divisible sequence,[7] and group cardinalities can take the values $k_1, \ldots, k_r$, for some constant $r \geq 1$. Such instances arise in our campaign scheduling application. Indeed, the most common lengths for TV ads are $15, 30$ and 60 seconds [22, 13]. Also, there are standard sizes of $150, 300$ and 600 pixels for web-banners on the Internet [21].

Finally, we present (in Appendix D) hardness results for the different all-or-nothing variants of GAP studied in the paper.

Technical Contribution: Our approximation algorithm for AGAP (in Section 2) uses a novel reduction of AGAP to maximizing submodular function subject to matroid constraint. At the heart of our reduction lies the fact that the sequence of sizes of large groups can be discretized to yield a logarithmic number of size categories. Thus, we can guarantee that the set of fractionally packed groups, in the initial Maximization Phase of the algorithm, has a total size at most m. Our reduction to submodular maximization encodes this knapsack constraint as a matroid constraint, by considering feasible vectors $(n_1, \ldots, n_H)$, where n_h gives the number of groups taken from size category h, for $1 \leq h \leq H$. These vectors (which are *implicitly* enumerated in polynomial time) are used for defining the matroid constraint.

Our definition of the submodular set function, $f(S)$ (see Section 2), which finds *fractional* packing of items, in fact guarantees that the rounding that we use for group sizes (to integral powers of $1+\varepsilon$, for some $\varepsilon > 0$), causes only small harm to the approximation ratio. This allows also to define a non-standard polynomial time implementation of an algorithm of [1], for maximizing a submodular function under matroid constraint. More precisely, while the universe for our submodular function f is of exponential size, we show that f can be computed in polynomial time.

Our randomized approximation scheme for AGAP instances with constant number of bins (in Section 3.1) is based on a non-trivial LP relaxation of the problem. While the resulting LP has polynomial size when the number of bins is fixed, solving it in polynomial time for general instances (where the number of variables is exponentially large) requires sophisticated use of separation oracles, which is of independent interest. The fractional solution obtained for the LP is rounded by using an approximation technique of [10, 9] for maximizing a submodular function subject to fixed number of knapsack constraints.

[7] A sequence $d_1 < d_2 < \cdots < d_z$ is a *divisible* if d_i divides d_{i-1} for all $1 < i \leq z$.

1.2 Related Work

All-or-nothing GAP generalizes several classical problems, including GAP (with same sizes across the bins), the *multiple knapsack problem* (MKP), *multiple knapsack with assignment restrictions* [15], (MKAR), and the *generalized multi assignment problem*. In this section we briefly summarize the state of the art for these problems.

As mentioned above, the special case where all groups consist of a *single* item yields an instance of GAP, where each item takes a single size over all bins. GAP is known to be APX-hard already in this case, even if there are only two possible item sizes, each item can take two possible profits, and all bin capacities are identical [2]. The best approximation ratio obtained for GAP is $\frac{e}{e-1} + \varepsilon$ [4].

In minimum GAP (see, e.g., [11]), there are m machines and n jobs. Each machine i is available for T_i time units, and each job has a processing time (size), and a cost of being assigned to a machine. The goal is to schedule all the jobs at minimum total cost, where each job needs to be assigned to a single machine. The paper [18] gives an algorithm which minimizes the total cost, using a schedule where each machine i completes within $2T_i$ time units, $1 \leq i \leq m$.

The generalized multi-assignment problem extends minimum GAP to include multiple assignment constraints. Job processing times and the costs depend on the machine to which they are assigned, the objective is to minimize the costs, and all the jobs must be assigned. The only differences from GAP are the multiple assignment constraints of each job. This problem was discussed in [16], where Lagrangian dual-based branch-and-bound algorithms were used for obtaining an exact solution for the problem.[8]

We are not aware of earlier work on AGAP or *all-or-nothing* variants of other packing problems.

2 Approximation Algorithm for AGAP

In this section we consider general AGAP instances, where each item ℓ has a size $s_\ell \in (0, 1]$ and arbitrary utilities across the bins. We assume throughout this section that all bins are of the same (unit) capacity. Our approach is based on a version of AGAP, called RELAXED-AGAP, obtained by relaxing the constraint that the total size of items packed in a bin must be at most 1, and by defining the *utility* of a solution to RELAXED-AGAP slightly differently. We prove that the maximum utility of a solution to RELAXED-AGAP upper bounds the objective value of the optimal AGAP solution. Our algorithm proceeds in two phases.

Maximization Phase: The algorithm approximates the optimal utility of RELAXED-AGAP in polynomial time, by applying a novel reduction to submodular function maximization under matroid constraints. Let S denote the subset of groups assigned by this RELAXED-AGAP solution.

Filling Phase: The algorithm next chooses a subset $S' \subseteq S$ whose utility is at least a constant fraction of the utility of S. Then, the algorithm constructs a feasible solution for AGAP that assigns the groups in S' (not necessarily to the

[8] The running time of this algorithm is not guaranteed to be polynomial.

same bins as the RELAXED-AGAP solution) and achieves AGAP value that is at least half of the utility of S', thereby obtaining $O(1)$-approximation for AGAP.

2.1 Maximization phase

RELAXED-AGAP: The input for RELAXED-AGAP is the same as that for AGAP. A feasible RELAXED-AGAP solution is a subset S of the groups whose total size is no more than m (the total size of the bins) and a *valid* assignment p of the items in groups in S to bins; a valid assignment is defined as one in which no two items from the same group are assigned to the same bin. In RELAXED-AGAP, we do not have a constraint regarding the total size of the items assigned to a single bin. Given a solution (S, p) and a bin $j \in [m]$, let $p^{-1}(j) \subseteq [N]$ be the set of items assigned by p to bin j. The utility of a solution (S, p) is the sum of the utility contributions of the bins. The utility contribution of a bin $j \in [m]$ is the maximum value from (fractionally) assigning items in $p^{-1}(j)$ to j satisfying its unit capacity. In other words, we solve for bin j the *fractional knapsack* problem. To define this more formally, we introduce some notation.

Definition 1. *Given a subset $I \subseteq [N]$ of items and a bin j, define $\pi(j, I) = \max_{\boldsymbol{w}} \sum_{\ell \in I} w_\ell a_{\ell j}$, where the maximum is taken over all weight vectors $\boldsymbol{w} \in \Re_+^{|I|}$ that assign weights $w_\ell \in [0, 1]$ to $\ell \in I$, satisfying $\sum_{\ell \in I} w_\ell s_\ell \leq 1$.*

It is easy to determine $\boldsymbol{w}$ that maximizes the utility contribution of bin j. Order the items in I as $\ell_1, \ldots, \ell_b$ in the non-increasing order of their ratio of utility to size, i.e., $a_{\ell_1 j}/s_{\ell_1} \geq a_{\ell_2 j}/s_{\ell_2} \geq \cdots \geq a_{\ell_b j}/s_{\ell_b}$. Let d be the maximum index such that $s = \sum_{i=1}^{d} s_{\ell_i} \leq 1$. Set $w_1 = \cdots = w_d = 1$. If $s < 1$ and $d < b$, set $w_{d+1} = (1 - s)/s_{\ell_{d+1}}$. Set the other weights $w_{d+2} = \cdots = w_b = 0$.

Using the above notation, the utility of a solution (S, p) is given by $\sum_{j \in [m]} \pi(j, p^{-1}(j))$. The RELAXED-AGAP is to find a solution with maximum utility.

We can extend Definition 1 to *multiset* I of $[N]$ as follows.

Definition 2. *Think of a multiset I of $[N]$ as a function $I : [N] \to \mathbb{Z}_+$ that maps each $\ell \in [N]$ to a non-negative integer equal to the number of copies of ℓ present in I. Define $\pi(j, I) = \max_{\boldsymbol{w}} \sum_{\ell \in [N]} w_\ell a_{\ell j}$, where the maximum is taken over all weight vectors $\boldsymbol{w} \in \Re_+^N$ that assign weights $w_\ell \in [0, I(\ell)]$ to $\ell \in [N]$ satisfying $\sum_{\ell \in [N]} w_\ell s_\ell \leq 1$.*

Solving RELAXED-AGAP near-optimally: Recall that a valid assignment of a subset of items in $[N]$ to bins is one in which no two items from a group get assigned to the same bin. Now define a universe U as follows:

$$U = \{(G, L) \mid L \text{ is a valid assignment of } \textit{all} \text{ items in group } G \text{ to bins } [m]\}$$

A subset $S \subseteq U$ defines a multiset of groups that appear as the first component of the pairs in S. Below, we use $G(S)$ to denote the multiset of such groups. For a subset $S \subseteq U$ and a bin $j \in [m]$, let $I_j = \uplus_{(G,L) \in S} L^{-1}(j)$ be the *multiset* union of sets of items mapped to j over all elements $(G, L) \in S$. Note that I_j can

indeed be a multiset since S may contain two elements (G_1, L_1) and (G_2, L_2) with $G_1 = G_2$. Now define

$$f(S) = \sum_{j \in [m]} \pi(j, I_j).$$

The following important but simple claim is proved in the Appendix.

Claim 1 *The function $f(S)$ is non-decreasing and submodular.*

To identify subsets $S \subset U$ that define feasible RELAXED-AGAP solutions, we need two constraints.

Constraint 1. The subset S does not contain two elements (G_1, L_1) and (G_2, L_2) such that $G_1 = G_2$.

Constraint 2. The total size of the groups in $G(S)$, counted with multiplicities, is at most m, i.e., $\sum_{(G,L)\in S} \sum_{\ell \in G} s_\ell \leq m$.

Constraint 1 is easy to handle since it is simply the independence constraint in a partition matroid. Unfortunately, Constraint 2, which is essentially a knapsack constraint, is not easy to handle over the exponential-sized universe U.

Handling Constraint 2 approximately in polynomial time: To this end, we split the groups into a logarithmic number of classes. Fix $\epsilon > 0$. Class 0 contains all groups G such that $s(G) := \sum_{\ell \in G} s_\ell \leq \varepsilon m/n$. For $h \geq 1$, class h contains all groups G with $s(G) \in (\varepsilon m/n \cdot (1+\varepsilon)^{h-1}, \varepsilon m/n \cdot (1+\varepsilon)^h]$. We use $\mathcal{C}_h$ to denote class h. Since $s(G) \leq m$ for all groups G, there are only $H = O(1/\varepsilon \cdot \log(n/\varepsilon))$ non-empty classes. We enforce an upper bound of m on the total size of groups in $G(S)$ by enforcing an upper bound on the total size of groups in $G(S)$ from each class separately. We call a vector $(y_1, \ldots, y_H) \in \mathbb{Z}_+^H$ of non-negative integers *legal* if $\sum_{h=1}^H y_h \leq H(1+1/\varepsilon)$. Note that the number of legal vectors is $O(\binom{H(1+1/\varepsilon)}{H}) = O(2^{H(1+1/\varepsilon)})$, which is polynomial in m and n.

Lemma 2. *For any $S \subseteq U$ satisfying Constraint 2, there exists a legal vector $(y_1, \ldots, y_H)$ such that for all $h \in [H]$, the number of groups in $G(S)$, counted with multiplicities, that are in $\mathcal{C}_h$ is at most $\hat{y}_h := \lfloor y_h n/(H(1+\varepsilon)^{h-1}) \rfloor$.*

This lemma implies, in particular, that the optimum solution to AGAP satisfies the above property as well. With this motivation, we define $U_h = \{(G, L) \in U \mid G \in \mathcal{C}_h\}$ and define a new constraint as follows.

Constraint 2′ for a fixed legal vector $(y_1, \ldots, y_H)$. For each $1 \leq h \leq H$, the number of the groups in $G(S)$, counted with multiplicities, that are in $\mathcal{C}_h$ is at most $\hat{y}_h$ as defined in Lemma 2.

Lemma 3. *Fix a legal vector $(y_1, \ldots, y_H)$. The collection of all $S \subseteq U$ satisfying Constraint 1 and Constraint 2′ for this vector defines a* laminar *matroid $M(y_1, \ldots, y_H)$ over U. Furthermore, an independent set $S \subseteq U$ in this matroid satisfies $\sum_{(G,L)\in S} \sum_{\ell \in G} s_\ell \leq m((1+\varepsilon)^2 + \varepsilon)$.*

Given a legal vector $(y_1, \ldots, y_H)$, consider a problem, called SUBMOD-MATROID, of maximizing the non-decreasing submodular function $f(S)$ over all independent sets in the matroid $M(y_1, \ldots, y_H)$. Recall that Nemhauser et al. [12] proved that a greedy algorithm that starts with an empty set and iteratively adds "most profitable" element to it while maintaining independence, as long as possible, is a 1/2-approximation. Each iteration can be implemented in polynomial time as follows. Given a current solution S and a group G, the problem of finding the assignment L that increases the utility f relative to S by the maximum amount can be cast as a bipartite matching problem. To see this, create a bipartite graph with elements in G as vertices on the left-hand-side and bins as vertices on the right-hand-side. For $\ell \in G$ and a bin j, add an edge (ℓ, j) with weight equal to the amount by which contribution of bin j would increase if ℓ is added to bin j. This quantity, in turn, can be computed by solving a fractional knapsack problem on bin j. The maximum weight assignment corresponds to the maximum-weight matching in this graph.

In the maximization phase, we enumerate over all (polynomially many) legal vectors and compute a 1/2-approximate solution to the corresponding SUBMOD-MATROID problem. In the end, we pick the maximum valued solution over all such solutions.

Improving the approximation to $(e-1)/e$: Instead of the greedy algorithm of Nemhauser et al. [12], we can also use the $\frac{e}{e-1}$-approximate Continuous Greedy Algorithm of Calinescu et al. [1]. Some care is needed to show that this algorithm can indeed to implemented in polynomial time in our setting. We omit the details from this version, however, due to lack of space.

In summary, we find a set $S^* \subseteq U$ such that (1) each group appears at most once in $G(S^*)$, (2) the total size of the groups in $G(S^*)$ is at most $m((1+\varepsilon)^2+\varepsilon) \leq m(1+4\varepsilon)$ (if $(\varepsilon \leq 1)$, and (3) $f(S^*)$ is at least 1/2 (or $(e-1)/e$ if we use the algorithm of Calinescu et al. [1]) of the maximum value achieved by such sets.

2.2 Filling phase

We show how to choose a subset of the groups in $G(S^*)$ and a feasible assignment of the items in these groups such that the utility of these assignments is a constant fraction of $f(S^*)$. In the description we use parameters $u, v > 0$, whose value will be optimized later.

Lemma 4. *Assume $v \geq 4$, $v(1+4\varepsilon) < u$ and $k_{\max} := \max_i k_i \leq m/2$. In polynomial time, we can compute a subset of groups $F \subseteq G(S^*)$ and a feasible assignment of their items to the bins, forming a feasible solution to* AGAP *with value at least $f(S^*) \cdot \min\{1/u, \frac{1}{2}(1/v(1+4\varepsilon) - 1/u)\}$.*

Recall that $f(S^*) = \sum_j \pi(j, I_j)$, where I_j is a set of items mapped to bin j over all $(G, L) \in S^*$. Since S^* satisfies Constraint 1, we do not have two elements $(G, L_1), (G, L_2) \in S^*$ for any G. We now subdivide the value $f(S^*)$ into the groups $G \in G(S^*)$, naturally, as follows. Suppose that $\pi(j, I_j)$ is achieved by a weight-vector $\boldsymbol{w}(j)$. Fix any such optimum weight vector $\boldsymbol{w}^*(j)$ for each j. These vectors, when combined, give a weight vector $\boldsymbol{w}^* \in \Re_+^N$, assigning a unique

weight w^*_ℓ to each $\ell \in [N]$. We define the *contribution* of a group $G \in G(S^*)$ to $f(S^*)$ as $\sigma^*(G) = \sum_{\ell \in G} w^*_\ell a_{\ell L(\ell)}$ where $(G, L) \in S^*$.

Proof of Lemma 4 If there is a group $G \in G(S^*)$ with $\sigma^*(G) \geq f(S^*)/u$, we output $F = \{G\}$ with the best assignment of items in G to bins (computed using maximum matching, as described in the previous section) as solution. Clearly, the utility of this solution is at least $f(S^*)/u$.

Suppose that no such group exists. In this case we consider the groups $G \in G(S^*)$ in non-increasing order of $\sigma^*(G)/s(G)$. Choose the longest prefix in this order whose total size is at most m/v. Let $T \subset S^*$ be the solution induced by these groups. We first argue that $T \neq \emptyset$. Note that T can be empty only if the first group G in the above order has size more than m/v. Thus $\sigma^*(G)/(m/v) > \sigma^*(G)/s(G) \geq f^*(S)/(m(1+4\varepsilon))$. The second inequality holds since the total size of groups in $G(S^*)$ is at most $m(1+4\varepsilon)$ and the "density" $\sigma^*(G)/s(G)$ of G is at least the overall density of $G(S)$, which in turn is at least $f^*(S)/(m(1+4\varepsilon))$. This implies that $\sigma^*(G) > f^*(S)/(v(1+4\varepsilon)) > f^*(S)/u$, a contradiction. We now show how to find a feasible solution to AGAP that consists of groups in $G(T)$ and whose value is at least $f(T)/2$. ■

The rest of the algorithm proceeds in three steps as given below.

1. Eliminate all zero weights: Let $\boldsymbol{w} \in \Re^N_+$ be the weight vector that determines the value $f(T)$. Note that the weight w_ℓ assigned to some of the items ℓ in groups in $G(T)$ may be zero. We modify the assignment of items in the solution T so that no item would have zero weight. Note that if an item ℓ assigned to bin j in solution S has $w_\ell = 0$, the total size of the items assigned to bin j in S is at least 1. It follows that there are at most $\lfloor m/v \rfloor$ bins that may contain items of zero weight, since the total size of all items assigned in T is no more than m/v.

For each item with zero weight that belongs to a group G_i, there is at least one bin j such that the total size of the items assigned to bin j is less than 1 and no items from group G_i are assigned to bin j. This follows since $|G_i| + \lfloor m/v \rfloor \leq m/2 + \lfloor m/v \rfloor < m$. It follows that this item can be assigned to bin j and be assigned non-zero weight. We can continue this process as long as there are items with zero weight, thereby, eliminating all zero weights.

2. Evicting overflowed items: Suppose there are a (respectively, b) bins that are assigned items of total size more than 1 (respectively, more than $1/2$ and at most 1). Call these bins 'full' (respectively, 'half full'). Since the total volume of packed items is at most m/v, we have $a + b/2 \leq m/v$. Next, we remove some items from these a full bins to make the assignment feasible. Consider such a bin. We keep in this bin either all the items assigned to it that have weight equal to 1, or the unique item that has weight strictly between 0 and 1, whichever contributes more to $f(T)$. In this step, we lose at most half of the contribution of the full bins to $f(T)$. We further evict all items assigned to the least profitable $\lfloor (m-a)/2 \rfloor$ non-full bins. In this step, we lose at most half of the contribution of the non-full bins to $f(T)$.

3. Repacking evicted items: We now repack all the evicted items to maintain feasibility of the solution. We first repack evicted items of size at least half. Note

that are at most a such items (from full bins) and at most b such items (from half full bins). These $a+b$ items can be packed into evicted $\lfloor (m-a)/2 \rfloor$ bins by ensuring $a+b \leq \lfloor (m-a)/2 \rfloor$, i.e., $3a+2b < m$. This is indeed true since $v \geq 4$ together with $a + b/2 \leq m/v$ implies $4a + 2b \leq m$.

We are now left only with items whose size is less than half to repack. For each such item from group i, we find a bin whose total size is less than half – that does not contain another item from group i – and insert the item to this bin. Note that, since the size of the item is less than half, the solution remains feasible. Since the total size of the items to be packed is at most m/v, there are at most $\lfloor 2m/v \rfloor$ bins of size at least half. Thus, we are guaranteed to find such a bin in case $m - \lfloor 2m/v \rfloor - k_i \geq 0$, i.e., $k_i \leq \lceil m(1-2/v) \rceil$.

We now bound $f(T)$. Since the contribution of any group to $f(S^*)$ is no more than $f(S^*)/u$, the contribution of the groups in T is at least $f(S^*) \cdot (1/v(1+4\varepsilon) - 1/u)$. Recall that the reduction in $f(T)$ due to the eviction of items is at most half of $f(T)$. Thus the value of the final solution is at least $f(S^*) \cdot \frac{1}{2}(1/v(1+4\varepsilon) - 1/u)$.

This completes the proof of Lemma 4.

Now to bound the overall approximation ratio, we would like $1/u = \frac{1}{2}(1/v(1+4\varepsilon) - 1/u)$. Hence we set $u = 3v(1+4\varepsilon)$. Thus, with $v = 4$ and $u = 12(1+4\varepsilon)$, we get a ratio of $\frac{1}{12(1+3\varepsilon)}$. Since we lost a factor of $1/2$ (or $(e-1)/e$) in the maximization phase, we get an overall $24(1+4\varepsilon)$-approximation (or $12(1+\varepsilon)\frac{e}{e-1}$-approximation).

This proves the following theorem.

Theorem 1. AGAP *admits a polynomial-time* $12(1+\varepsilon)\frac{e}{e-1}$*-approximation for any* $0 < \epsilon < 1$ *provided any group has at most* $k_{\max} \leq m/2$ *items.*

3 Approximating Special Cases of AGAP

In this section we consider several special cases of AGAP. We assume throughout the discussion that the bins have uniform (unit) capacities.

3.1 LP Based Approximation Scheme for Constant Number of Bins

We formulate the following LP relaxation for AGAP. For every group G_i, we define $\mathcal{P}_i$ to be the collection of admissible packings of elements of group G_i alone. The relaxation has an indicator variable $x_{i,p}$ for every group G_i and admissible assignment $p \in \mathcal{P}_i$. Beside the constraints of AGAP, we further require the total size of the elements in the fractional solution to be at most $M \in [0, m]$. Note that this LP is a relaxation of AGAP only for $M = m$.

$$
\begin{array}{lll}
\text{(AGAP-LP)} \max & \sum_{i\in[n]} \sum_{p\in\mathcal{P}_i} \left(x_{i,p} \cdot \sum_{\ell\in G_i} a_{\ell p(\ell)} \right) & \\
\text{s.t.} & \sum_{p\in\mathcal{P}_i} x_{i,p} & \leq 1 \quad \forall\, i \in [n] \qquad (1) \\
& \sum_{i\in[n]} \sum_{p\in\mathcal{P}_i \mid \exists\, \ell : p(\ell) = j} x_{i,p} \cdot s_\ell & \leq c_j \quad \forall\, j \in [m] \qquad (2) \\
& \sum_{i\in[n]} \sum_{p\in\mathcal{P}_i} \left(x_{i,p} \cdot \sum_{\ell\in G_i} s_\ell \right) & \leq M \qquad (3) \\
& x_{i,p} & \geq 0 \quad \forall\, i \in [n], p \in \mathcal{P}_i
\end{array}
$$

Constraint (1) requires every group to have at most one assignment. Constraint (2) guarantees that no bin is over-packed. Finally, constraint (3) enforces that the total size of the packed elements does not exceed M.

Lemma 5. *AGAP-LP can be solved in polynomial time.*

The proof of Lemma 5 is based on finding a separation oracle for the dual LP (see in Appendix E).

We present an approximation scheme for the case where the number of bins is a constant. The algorithm uses AGAP-LP, which is this case is of polynomial size and thus can be solved in polynomial time using standard techniques. However, we apply a different rounding procedure to the solution of this LP. This rounding procedure draws many ideas from the rounding procedure suggested in [10, 9] for the problem of maximizing a submodular function subject to a constant number of knapsack constraints.

The idea of the rounding procedure is to guess the most valuable groups of the optimal solution and their corresponding assignment in this solution. Note that this can be done efficiently because the number of bins is constant. None of the remaining groups can be valuable on their own, and therefore, we can safely dismiss all such groups containing a large element. This allows us to show, via concentration bounds, that a randomized rounding satisfies the capacity constraints of all bins with high enough probability (recall that all remaining elements are small). We give the details in Appendix A.

Theorem 2. *There is a randomized polynomial time approximation scheme for* AGAP *with fixed number of bins.*

3.2 Approximation Algorithm for Unit Size Items

In the special case where all items have unit sizes, we give the best possible approximation ratio.

Theorem 3. AGAP *with unit item sizes admits an* $\frac{e}{e-1}$*-approximation.*

3.3 The All-or-Nothing Group Packing Problem

For AGAP instances where each group G_i has a utility $P_i > 0$ if all of its items are packed, and 0 otherwise, we show that AGAP can be approximated within a small constant $\rho \in (2, 3 + \varepsilon]$, for some $\varepsilon > 0$. Specifically,[9]

Theorem 4. *There is a* $(\frac{2(\gamma+1)}{\gamma} + \varepsilon)$*-approximation for all-or-nothing group packing, where* $\gamma = \lfloor \frac{m}{k_{\max}} \rfloor$.

4 Polynomial Time Approximation Schemes for AAP

In Appendix C we give approximation schemes for two subclasses of the all-or-nothing assignment problem which are of practical interest.

[9] We give the details in Appendix B.

References

1. G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. *SIAM J. on Computing*, 40(6), 2011.
2. C. Chekuri and S. Khanna. A PTAS for the multiple knapsack problem. *SIAM J. on Computing*, 35(3):713–728, 2006.
3. V. Dureau. Addressable advertising on digital television. In *Proceedings of the 2nd European conference on interactive television: enhancing the experience*, Brighton, UK, March–April 2004.
4. U. Feige and J. Vondrák. Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e. In *FOCS*, pages 667–676, 2006.
5. M. R. Garey and D. S. Johnson. *Computers and In-tractability. A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York, January 1979.
6. Google AdWords. `http://adwords.google.com`.
7. V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37(1):27–35, 1991.
8. E. M. Kim and S. S. Wildman. A deeper look at the economics of advertiser support for television: the implications of consumption-differentiated viewers and ad addressability. *Journal of Media Economics*, 19:55–79, 2006.
9. A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. full version. `http://www.cs.technion.ac.il/~hadas/PUB/max_submodular.pdf`.
10. A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *SODA*, pages 545–554, 2009.
11. O. E. Kundakcioglu and S. Alizamir. Generalized assignment problem. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 1153–1162. Springer, 2009.
12. G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Math. Programming*, 14:265–294, 1978.
13. Nielsen Media Research. Advertising fact sheet. blog.nielsen.com, September 2010.
14. Nielsen Media Research. Three Screen Report - Q4 2010 - Volumne 8. blog.nielsen.com, September 2010.
15. Z. Nutov, I. Beniaminy, and R. Yuster. A (1-1/e)-approximation algorithm for the generalized assignment problem. *Oper. Res. Lett.*, 34(3):283–288, 2006.
16. J. Park, B. Lim, and Y. Lee. A Lagrangian dual-based branch-and-bound algorithm for the generalized multi-assignment problem. *Management Science*, 44:271–282, 1998.
17. K. Pramataris, D. Papakyriakopoulos, G. Lekakos, and N. Mulonopoulos. Personalized Interactive TV Advertising: The iMEDIA Business Model. *Electronic Markets*, 11:1–9, 2001.
18. D. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62(1):461–474, 1993.
19. SintecMedia - On Air. `http://www.sintecmedia.com/OnAir.html`.
20. M. Sviridenko. A note on maximizing a submodular set function subject to knapsack constraint. *Operations Research Letters*, 32:41–43, 2004.
21. The Interactive Advertising Bureau (IAB). `http://iab.net`.
22. C. Young. Why TV spot length matters. *Admap*, (497):45–48, September 2008.

A Approximation Scheme for Constant Number of Bins

In this section, we present an approximation scheme for the case where the number of bins is considered a constant. The algorithm uses AGAP-LP which was defined in Section 3.1. The rounding procedure draws many ideas from the rounding procedure suggested by [10, 9] for the problem of maximizing a submodular function subject to a constant number of knapsack constraints.

Given a group G_i which is packed by OPT, let p'_i denote the packing of the elements of G_i induced by OPT. For two packings p_1 and p_2 which pack a disjoint set of elements (i.e., for every $\ell \in [N]$, either $p_1(\ell) = \perp$ or $p_2(\ell) = \perp$). Let $p_1 \odot p_2$ denote the union of the two packings. Formally, if $p = p_1 \odot p_2$, then for every $\ell \in [N]$:

$$p(\ell) = \begin{cases} p_1(\ell) \text{ if } p_1(\ell) \neq \perp \\ p_2(\ell) \text{ if } p_2(\ell) \neq \perp \\ \perp \quad \text{otherwise} \end{cases} .$$

Using the above notation, we present Algorithm 1 for AGAP. The algorithm gets a single parameter ε which determines its approximation ratio: the smaller ε, the better the approximation factor (and the worse the time complexity). The rest of this section is devoted for analyzing this algorithm.

Lemma 6. *For a constant ε, Algorithm 1 has a polynomial time complexity.*

Proof. It is clear that all steps beside Lines 1 and 2 can be performed in polynomial time. Let us show that these lines can also be performed in polynomial time. Lines 1 and 2 together guess up to $\varepsilon^{-4}m^3$ pairs (G_i, p'_i), where $p'_i \in \mathcal{P}_i$. Notice that for every group G_i:

$$|\mathcal{P}_i| \leq \frac{m!}{(m - |G_i|)!} \leq m! = O(1) .$$

Hence, the number of possible pairs is only:

$$\sum_{i \in [n]} |\mathcal{P}_i| = O(n) .$$

The algorithm is required to guess only a constant number of pairs, which leaves only a polynomial number of possible guesses. ∎

Algorithm 1 dismisses large assignments of groups which does not belong to $\mathcal{A}$. We would like to show that this does not effect the approximation ratio by much. Let $\mathcal{P}_B$ be the collection of large assignments of groups outside of $\mathcal{A}$ which agree with OPT. Formally,

$$\mathcal{P}_B = \{p'_i | i \notin \mathcal{A} \text{ and } p'_i \text{ is large}\} .$$

The following lemma upper bounds the total revenue of all packings of $\mathcal{P}_B$.

Lemma 7. *The total revenue of the packings in $\mathcal{P}_B$ is at most $\varepsilon \cdot OPT$. Formally, $\sum_{p'_i \in \mathcal{P}_B} \sum_{\ell \in G_i} a_{\ell p'_i(\ell)} \leq \varepsilon \cdot OPT$.*

Algorithm 1 Algorithm for Constant Number of Bins(ε)

Guess the collection $\mathcal{A}$ of the $\varepsilon^{-4}m^3$ most valuable groups of OPT.
Guess for every set $G_i \in \mathcal{A}$ the packing p'_i.
Remove all groups of $\mathcal{A}$ from AGAP-LP.
for every bin $j \in [m]$ **do**
  Update the capacity of bin j in AGAP-LP to $r_j = c_j - \sum_{G_i \in \mathcal{A} | \exists\, \ell \in G_i : p'_i(\ell) = j} s_\ell$.
end for
An assignment $p \in \mathcal{P}_i$ of a group $G_i \notin \mathcal{A}$ is large if there exists an element $\ell \in G_i$ such that $s_\ell \geq \varepsilon^3 m^{-2} r_{p(\ell)}$.
Remove from AGAP-LP all variables $x_{i,p}$ corresponding to large assignments.
Solve the resulting LP, let y be the solution found.
Let $P = \bigodot_{G_i \in \mathcal{A}} p'_i$ (i.e., P is the result of applying $\odot$ to all the packings p'_i corresponding to groups $G_i \in \mathcal{A}$).
for every group $G_i \notin \mathcal{A}$ **do**
  Randomly select at most one packing $p''_i \in \mathcal{P}_i$, where the probability of every packing p''_i is $(1-\varepsilon) \cdot y_{i,p''_i}$.
  if some packing p''_i was selected **then**
    $P = P \odot p''_i$.
  end if
end for
if U is an admissible packing **then**
  Return P.
else
  Return an empty packing (i.e., a packing assigning all elements to $\perp$).
end if

Proof. First, let us upper bound $|\mathcal{P}_B|$. Since all packings $\mathcal{P}_B$ agree with OPT, which is an admissible solution, we get for every bin $j \in [m]$:

$$\sum_{G_i \in \mathcal{A} | \exists\, \ell \in G_i : p'_i(\ell) = j} s_\ell \quad + \sum_{p'_i \in \mathcal{P}_B | \exists\, \ell \in G_i : p'_i(\ell) = j} s_\ell \quad \leq c_j \ .$$

By definition, the first sum is $c_j - r_j$. The second term can be lower bounded by: $|\mathcal{P}_{B,j}| \varepsilon^3 m^{-2} r_i$, where $\mathcal{P}_{B,j}$ is the subset of $\mathcal{P}_B$ containing only packings assigning an element of size over $\varepsilon^3 m^{-2} r_j$ to B_j. Plugging both observations into the previous inequality gives:

$$|\mathcal{P}_{B,j}| \varepsilon^3 m^{-2} r_j \leq r_j \Rightarrow |\mathcal{P}_{B,j}| \leq \varepsilon^{-3} m^2 \ .$$

Since $\mathcal{P}_{B,j}$ is a subset of $\mathcal{P}_B$ for every bin $j \in [m]$, we can use the union bound to get:

$$|\mathcal{P}_B| \leq \sum_{j \in [m]} |\mathcal{P}_{B,j}| \leq \varepsilon^{-3} m^3 \ .$$

Next, we upper bound the total revenue of the packings in $\mathcal{P}_B$. Each such packing $p \in \mathcal{P}_B$ was not guessed by Line 2, despite the fact that $p = p'_i$ for some group G_i. Hence, the revenue of p is either equal or lower than the revenue of all packings in $\mathcal{P}_A = \{p'_i | G_i \in \mathcal{A}\}$. On the other hand, the total revenue of all packings in $\mathcal{P}_A$ is at most OPT, and therefore, there exists a packing in $\mathcal{P}_A$ with value of $OPT/|\mathcal{A}| = \varepsilon^4|\mathcal{B}|^{-3} \cdot OPT$ or less. Thus, the revenue of p is at most $\varepsilon^4|\mathcal{B}|^{-3} \cdot OPT$.

The lemma now follows from multiplying the upper bound on the number of packings in $\mathcal{P}_B$ with the upper bound on the revenue from each such packing. ■

Let R_A denote the total revenue of the packings in $\{p'_i | G_i \in \mathcal{A}\}$. Formally, $R_A = \sum_{G_i \in \mathcal{A}} \sum_{\ell \in G_i} a_{\ell p'_i(\ell)}$.

Corollary 1. *The solution of y of the LP produced by Line 9 of Algorithm 1 has value of at least $(1-\varepsilon)OPT - R_A$.*

Proof. Let us construct an admissible packing OPT'. We start with OPT, and remove all elements belonging to a group of $\mathcal{A}$ or packed by some packing in $\mathcal{P}_B$. Notice that OPT' pack every group $G_i \notin \mathcal{A}$ with a packing p such that the variable $x_{i,p}$ is not removed by Algorithm 1. Thus, OPT' induces a feasible solution for the LP that algorithm solves on Line 9.

The corollary now follows since, by Lemma 7, OPT' has a revenue of at least $(1-\varepsilon)OPT - R_A$. ■

Let R_P denote the total revenue from the packing P. The following corollary lower bounds the expectation of R_P.

Corollary 2. $\mathbb{E}[R_P] \geq (1-2\varepsilon)OPT$.

Proof. Every packing p of a group $G_i \notin \mathcal{A}$ gets into P with probability $(1-\varepsilon) \cdot y_{i,p}$. By the linearity of the expectation, the expected contribution of these packings to R_P is:

$$\sum_{G_i \notin \mathcal{A}} \sum_{p \in \mathcal{P}_i} \left[(1-\varepsilon) \cdot y_{i,p} \cdot \sum_{\ell \in G_i} a_{\ell p(\ell)} \right] = (1-\varepsilon) \cdot \sum_{G_i \notin \mathcal{A}} \sum_{p \in \mathcal{P}_i} \left[y_{i,p} \cdot \sum_{\ell \in G_i} a_{\ell,p(\ell)} \right] \quad ,$$

which is exactly $1-\varepsilon$ times the value of y. By Corollary 1, the value of y is at least $(1-\varepsilon)OPT - R_A$. Hence, the groups of $\{G_i | i \in [n] \wedge G_i \notin \mathcal{A}\}$ contribute to R_P, in expectation, at least $(1-\varepsilon)^2 OPT - (1-\varepsilon)R_A \geq (1-2\varepsilon)OPT - R_A$.

Notice that the contribution of the groups of $\mathcal{A}$ to R_P is exactly R_A. The lemma now follows by summing the expected contributions of both types of groups to R_P. ■

Algorithm 1 outputs either P or an empty packing. The expected value of P was calculated in Corollary 2. To complete the analysis of Algorithm 1 we need to bound the expected value of P in the cases where the algorithm outputs an empty packing instead of P. Let Q_j be the ratio $c_j^{-1} \cdot \sum_{\ell \in [N] | P(\ell) = j} s_\ell$. We also define $Q = \max_{j \in [m]} Q_j$.

Lemma 8. *For every bin $j \in [m]$ and $h \geq 1$, $\Pr[Q_j \geq h] \leq \varepsilon m^{-2} h^{-2}$.*

Proof. After the packings of the groups of $\mathcal{A}$ are added to P, the remaining capacity of bin j is r_j. Hence, it will be enough if we upper bound the probability that the total size of the elements outside of the groups of $\mathcal{A}$ added to bin j exceeds hr_j. Let E be the set of elements outside of the groups of $\mathcal{A}$ that have a positive probability to be packed by P into bin j. For every element $\ell \in E$ we denote by q_ℓ its probability of ℓ to be packed by P into bin j, and denote by X_ℓ an indicator for the event that ℓ was indeed packed by P into bin j. We can now rephrase what we need to prove as $\Pr[\sum_{\ell \in E} s_\ell X_\ell \geq hr_j] \leq \varepsilon m^{-2} h^{-2}$.

Let us recall the information that we have on the elements of E.

- Since the probability of each element to be packed in bin j is determined by the solution y of AGAP-LP, we can conclude that $\sum_{\ell \in E} s_\ell q_\ell \leq (1-\varepsilon) r_j$.
- Since large packings were removed, for every $\ell \in E$, it must hold that $s_\ell \leq \varepsilon^3 m^{-2} r_j$.
- For every two elements ℓ_1, ℓ_2 belonging to different groups, X_{ℓ_1} and X_{ℓ_2} are independent.
- For every two elements ℓ_1, ℓ_2 belonging to a single group, X_{ℓ_1} and X_{ℓ_2} are mutually exclusive (i.e., $X_{\ell_1} + X_{\ell_2} \leq 1$).

The first observation states that the expected value of the sum $\sum_{\ell \in E} s_\ell X_\ell$ is at most $(1-\varepsilon) r_j$. Let us use the last three observations to lower bound the variance of this sum.

$$\begin{aligned} V\left[\sum_{\ell \in E} s_\ell X_\ell\right] &= \sum_{i \in [n]} V\left[\sum_{\ell \in G_i \cap E} s_\ell X_\ell\right] \leq \sum_{i \in [n]} \mathbb{E}\left[\left[\sum_{\ell \in G_i \cap E} s_\ell X_\ell\right]^2\right] \\ &= \sum_{i \in [n]} \mathbb{E}\left[\sum_{\ell \in G_i \cap E} s_\ell^2 X_\ell\right] = \sum_{\ell \in E} s_\ell^2 q_\ell \leq \max_{\ell \in E} s_\ell \cdot \sum_{\ell \in E} s_\ell q_\ell \\ &\leq [\varepsilon^3 m^{-2} r_j] \cdot r_j = \varepsilon^3 m^{-2} r_j^2 \ . \end{aligned}$$

We are now ready to bound $\Pr[\sum_{e_k \in E} s_k X_k > r_i]$ using Chebyshev's inequality.

$$\begin{aligned} \Pr\left[\sum_{\ell \in E} s_\ell X_\ell > hr_j\right] &\leq \Pr\left[\left|\sum_{\ell \in E} s_\ell X_\ell - \mathbb{E}\left[\sum_{\ell \in E} s_\ell X_\ell\right]\right| > h\varepsilon r_j\right] \leq \frac{\varepsilon^3 m^{-2} r_j^2}{(h\varepsilon r_j)^2} \\ &= \varepsilon m^{-2} h^{-2} \ . \end{aligned}$$

■

Corollary 3. *For every $h \geq 1$, $\Pr[Q \geq h] \leq \varepsilon m^{-1} h^{-2}$.*

Proof. Follows from Lemma 8 and the union bound. ■

Lemma 9. *If $Q \leq h$ for some $h > 1$, then $R_P \leq 4mh \cdot OPT$.*

Proof. P can be partitioned into $2mh$ admissible packings in the following manner. We order the groups packed by P arbitrarily. The maximum prefix of groups that form an admissible packing becomes the first packing in the partition. We then remove from P the groups of this prefix and repeat till P becomes empty. Notice that if some prefix cannot be extended due to the capacity constraint of bin j, then one of following two conditions must hold:

- The elements packed by the prefix into bin j have total size of at least $0.5c_j$.
- The next group in the order has an element of size at least $0.5c_j$ which is assigned by P to bin j.

Hence, after every two prefixes are removed, the total size of the elements packed into one bin j must decrease by at least $0.5c_j$. Since the total size of the elements packed into bin j by P is at most hc_j, the above procedure cannot be repeated more than $4mh$ times. ■

We are now ready to prove the approximation ratio of the algorithm.

Theorem 5. *Algorithm 1 is an* $(1-18\varepsilon)$*-approximation algorithm.*

Proof. The expected revenue of Algorithm 1 is $\mathbb{E}[R_P|Q \le 1]$. Let us lower bound this quantity as following:

$$\begin{aligned}\mathbb{E}[R_P|Q \le 1] &= \frac{\mathbb{E}[R_P] - \sum_{h=1}^{\infty} \Pr[2^{h-1} < Q \le 2^h] \cdot \mathbb{E}[R_P|2^{h-1} < Q \le 2^h]}{\Pr[Q \le 1]} \quad (1)\\ &\ge \mathbb{E}[R_P] - \sum_{h=1}^{\infty} \Pr[2^{h-1} < Q \le 2^h] \cdot \mathbb{E}[R_P|2^{h-1} < Q \le 2^h] \ .\end{aligned}$$

By Corollary 2, $\mathbb{E}[R_P] \ge (1-2\varepsilon)OPT$. By Corollary 3, $\Pr[2^{h-1} < Q \le 2^h] \le \Pr[2^{h-1} \le Q] \le \varepsilon m^{-1} 2^{2-2h}$. Finally, by Lemma 9, $\mathbb{E}[R_P|2^{h-1} < Q \le 2^h] \le 4m2^h \cdot OPT$. Combining all these observations into (1) gives:

$$\begin{aligned}\mathbb{E}[R_P|Q \le 1] &\ge (1-2\varepsilon)OPT - \sum_{h=1}^{\infty} (\varepsilon m^{-1} 2^{2-2h}) \cdot (4m2^h \cdot OPT)\\ &= (1-2\varepsilon)OPT - 16\varepsilon \cdot OPT \cdot \sum_{h=1}^{\infty} 2^{-h} = (1-18\varepsilon)OPT \ .\end{aligned}$$

■

B The All-or-Nothing Group Packing Problem

Consider the special case of AGAP where each group G_i has a utility $P_i > 0$ if all of its items are packed, and 0 otherwise. Alternatively, the utility of each item $\ell \in G_i$, when packed in bin $j \in [m]$, is $a_{\ell j} = \frac{P_i}{k_i}$. Let $S_i = \sum_{\ell \in G_i} s_\ell$ be the total size of G_i, $1 \le i \le n$. We assume throughout the discussion that $k_{\max} \le \frac{m}{2}$.

We give below a $(3+\varepsilon)$-approximation algorithm for the problem. Our bound (in Theorem 4), is given as $f(\alpha)$, where $\alpha = \frac{k_{\max}}{m}$. As α becomes small, the bound gets close to 2.

Algorithm 2 GroupPack

1: Run an FPTAS for Knapsack to find a subset of groups of maximum total utility, U_1, whose size is at most $\frac{m}{2}$.
2: Let U_2 be the total utility of the $\gamma = \lfloor \frac{1}{\alpha} \rfloor$ most profitable groups.
3: Output the solution of utility $\max\{U_1, U_2\}$.

We restate Theorem 4.

Theorem 4. *Algorithm GroupPack yields a* $(\frac{2(\gamma+1)}{\gamma} + \varepsilon)$*-approximation for all-or-nothing group packing.*

We use in the proof the next lemmas.

Lemma 10. *For any* $k_{\max} \geq 1$*, any subset of groups of total size* $W > 0$ *can be packed in at most* $\max\{2W, W + k_{\max}\}$ *unit-sized bins.*

Proof. Given a set of groups $G_1, \ldots, G_t$ of total size W, consider the following *balanced coloring* of the groups. We color the items of G_1 in arbitrary order, using at most $k_{\max}$ colors (so that each item receives a distinct color). Now, sort the items in G_2 in non-increasing order by size. Scanning the sorted list, we add the next item in G_2 to the color class of minimum total size. Similarly, we add to the color classes the items in $G_3, \ldots, G_t$. Let W_i denote the total size of color class i, where $1 \leq i \leq k_{\max}$. We note that, for any two color classes i, j, it holds that $|W_i - W_j| \leq 1$. Clearly, we can pack any color class $1 \leq i \leq k_{\max}$, whose total size is $W_i > 1$, in at most $2W_i$ unit sized bins (using FirstFit).

Now, to complete the proof, we distinguish between two cases for the sizes of the color classes.

(i) If for all $i \geq 1$, $W_i \geq \frac{1}{2}$, then either we can pack color class i in a single bin (if $W_i \leq 1$), or, we can pack it in at most $2W_i$ unit sized bins. Since $W = \sum_i W_i$, it follows that, in this case we can pack each color class in a separate set of bins, using at most $2W$ bins.
(ii) If there exists a color class i of total size $W_i < \frac{1}{2}$, then since we use a balanced coloring, for any color class $j \geq 1$, it holds that $W_j \leq \frac{3}{2}$. Let c_1 denote the number of color classes i of total size $W_i < 1$, then for each of these color classes we can use a single bin. Let c_2 be the number of color classes i whose total size is $W_i \in (1, \frac{3}{2}]$. For these color classes we use at most $2c_2$ bins. Overall, we can pack all the color classes in at most $c_1 + 2c_2 = (c_1 + c_2) + c_2 \leq k_{\max} + W$ bins.

■

Lemma 11. *Given an input for Group Packing, with* $\alpha \in (0, 1/2]$*, let* $c > 1$ *be some constant. Then one of the following holds.* (*i*) *There exist* γ *groups whose*

total utility is at least $\frac{OPT}{c}$*, or* (ii) *There exists a set of groups of total utility in* $[OPT(\frac{1}{2}-\frac{1}{\gamma c}), \frac{OPT}{2})$*, and whose total size is at most* $\frac{m}{2}$*.*

Proof. Assume that (i) does not hold, then we show that (ii) holds. Consider the set of groups in the instance. Sort these groups in non-increasing order of their utilities. Let L denote the sorted list. We now partition L to sub-lists. Scanning the list L from the first group, we close the first sub-list, L_1, when the total utility of the groups in this sub-list exceeds for the first time $OPT(\frac{1}{2}-\frac{1}{\gamma c})$. Similarly, we define the sub-lists $L_2, L_3, ...$

For $r \geq 1$, consider the last group, $G_{r\ell}$, added to L_r. Then, before $G_{r\ell}$ was added, the total utility of the group in L_r was smaller than $OPT(\frac{1}{2}-\frac{1}{\gamma c})$. Assume, by way of contradiction, that after we add $G_{r\ell}$ the total utility is at least $\frac{OPT}{2}$; then, the total utility of $G_{r\ell}$ is at least $\frac{OPT}{\gamma c}$. Since $c > 1$, there are at least γ groups in L_r, and since we added the groups non-increasing order by utilities, each of these γ groups has utility $\frac{1}{\gamma c}$ or larger. Contradiction.

Hence, the total utility of each sub-list is in $[OPT(\frac{1}{2}-\frac{1}{\gamma c}), \frac{OPT}{2})$. Therefore, there are at least 2 sub-lists. It follows, that there exists a sub-list whose total size is at most $\frac{m}{2}$. ■

Proof of Theorem 4: We first note that the selected groups can be feasibly packed. Indeed, if we select the groups in Step 2, then we can pack each group G_i in a separate set of at most αm bins, assigning a single item to each bin. Otherwise, we select the set of groups output by the FPTAS, in Step 1. By Lemma 10, we can pack these groups, whose total size is at most $\frac{m}{2}$, in at most m bins.

For the approximation ratio, let OPT denote the total utility of an optimal solution, and $c > 1$ some constant (to be determined). By Lemma 11, there exist a subset of groups of total utility at least $\min\{\frac{OPT}{c}, OPT(\frac{1}{2}-\frac{1}{\gamma c})\}$. Since in Step 1 we find a subset of groups of total utility at least $OPT(\frac{1}{2}-\frac{1}{\gamma c})(1-\varepsilon)$, taking $c = \frac{2(\gamma+1)}{\gamma}$, we have the statement of the theorem. ■

We note that when all groups are small, i.e., $S_i \leq \varepsilon m$, for some $\varepsilon > 0$, we can slightly modify the analysis of GroupPack to obtain the following.

Corollary 4. *If* $S_i \leq \varepsilon m$ *for some* $\varepsilon > 0$*, for all* $1 \leq i \leq n$*, then GroupPack is a* $(2+\varepsilon)$*-approximation algorithm, for any* $k_{max} \leq \frac{m}{2}$*.*

C Polynomial Time Approximation Schemes for AAP

Recall that in the All-or-nothing Assignment Problems, the items form n groups, where each group G_i, $1 \leq i \leq n$ is associated with a size s_i and a profit a_i, such that for every item $\ell \in G_i$, it holds that $s_\ell = s_i$ and $a_{\ell j} = a_i$ for all bins j. Denote by P_i the profit from satisfying G_i, that is, $P_i = k_i \cdot a_i$. Our PTASs consist of two steps:

(i) Guessing the set of groups to be satisfied.
(ii) Assigning the items of these groups to bins.

The first step is similar to the 'guessing-items' step in the PTAS proposed by Chekuri and Khanna for the Multiple Knapsack Problem [2]. The second step is implemented in different ways according to our assumptions on the instance. Different implementations are suggested for instances with unit-size items and for instances in which the item sizes form a divisible sequence and the groups have bounded cardinality.

Algorithm 3 PTAS's Overview

1: Guess the overall profit $\mathcal{O}$ such that $(1-\varepsilon)OPT \leq \mathcal{O} \leq OPT$.
2: Ignore all groups for which $P_i \leq \varepsilon\mathcal{O}/n$.
3: Scale all profits by $\varepsilon\mathcal{O}/n$ such that after the scaling all the profits are in the range $[n/\varepsilon]$.
4: Round down the profits to the nearest power of $(1+\varepsilon)$.
5: Divide the groups into $h \leq 3\ln n/\varepsilon$ distinct profit sets $S_1, \ldots, S_h$.
6: Guess the overall profit from each profit set in some optimal solution $U = \{U_1, \ldots, U_h\}$, i.e. guess a vector $(w_1, \ldots w_h)$ such that $w_i \in [h/\varepsilon^2]$ and $w_i(\varepsilon^2\mathcal{O}/h) \leq P(U_i) \leq (w_i+1)(\varepsilon^2\mathcal{O}/h)$.
7: Select the groups to be packed.
8: Pack the selected groups.

Whenever guessing is used, the algorithm actually performs a search over a polynomial-size scope. Using rounding and enumeration techniques similar to those used in the PTAS for MKP [2], it can be seen that the overall guessing process requires $O(\ln n/\varepsilon) \cdot O(n^{O(1/\varepsilon^3)})$ time and the total loss is bounded by factor of $(1-O(\varepsilon))$.

Instances with Unit-Size Items: Following steps (1) - (6) in **Algorithm 3**, we know the profit from each profit set, i.e., the tuple $(w_1, w_2, \ldots, w_h)$, that represents the profit from the h profit sets $S_1, \ldots, S_h$. The packing step is described in **Algorithm 4**. We use a greedy selection algorithm that picks the smallest groups, i.e. the groups with the smallest k_i values, as the groups to be satisfied (lines 1-5). Since all items have the same unit-size, a simple exchange argument implies that such a greedy choice is optimal.

The items of the selected subset of groups are packed using a greedy packing strategy (lines 7-9). Denote by $(\gamma_1^i, \ldots, \gamma_m^i)$ the remaining capacity of the bins before packing the items of the i-th group. Initially, for all bins $j \in [1..m]$ it holds that $\gamma_j^1 = c_j$. The groups are considered in arbitrary order. Group i is assigned to the k_i bins with the highest γ_j^i values. Ties are broken arbitrarily.

Algorithm 4 GreedySelectPack

1: **for** $i \in [1..h]$ **do**
2: Sort the groups in S_i in increasing order of cardinality.
3: Add groups to $Selected_i$ until the cumulative profit is in the range $[w_i(\varepsilon^2\mathcal{O}/h), (w_i+1)(\varepsilon^2\mathcal{O}/h)]$.
4: **end for**
5: $Groups' = \bigcup_{i\in[1..h]} Selected_i$.
6: **for** $G_i \in Groups'$ **do**
7: Assign G_i items to the k_i bins with the highest γ_j^i values.
8: **end for**

The correctness of our PTAS relies on the following theorem.

Theorem 6. *There exists a valid tuple $(w_1, w_2, \ldots, w_h)$ where each $w_i \in [h/\varepsilon^2]$ and $\sum_i w_i \leq \lceil h/\varepsilon^2 \rceil$ such that GreedySelectPack considers this tuple, selects a set $Groups'$ whose profit is $(1-\varepsilon)OPT$, and feasibly assign $Groups'$ to bins.*

Combining the above theorem with the analysis of the guessing-groups step we have:

Corollary 5. *APP with unit-size items admits a PTAS.*

Instances with Bounded-Cardinality and Item Sizes that Form a Divisible Sequence: A sequence $d_1 < d_2 < \cdots < d_z$ is a *divisible sequence* if d_{i-1} divides d_i for all $1 < i \leq z$. We present a PTAS for AAP in which the different item sizes $\{s_i\}$ form a divisible sequence, and z, the number of different size values, might be arbitrary. This type of instance arises in our applications as mentioned in Section 1.1. We consider instances with no assignment restrictions, i.e., all items from all groups can be assigned to all bins. In addition, we assume that there is a constant number of different group cardinalities, i.e. the cardinality values are $k_1, \ldots, k_r$, for some constant r.

In the guessing step, after guessing the optimal profit $\mathcal{O}$, we guess the matrix $(w_{11}, \ldots w_{1c}, \ldots, w_{h1}, \ldots w_{hr})$ for $1 \leq i \leq h$ and $1 \leq j \leq r$, such that $w_{ij} \in [h/\varepsilon^2]$ and $w_{ij}(\varepsilon^2\mathcal{O}/h) \leq P(U_i) \leq (w_{ij}+1)(\varepsilon^2\mathcal{O}/h)$. Let S_{ij} denote the set of groups in profit set i and cardinality k_j, where w_{ij} represents the total profit from groups in S_{ij}. The following greedy **Algorithm 5** is used to select the subset of groups to be assigned. The idea is to pick the groups with the smallest s_i values among each set. Recall that γ_j^i is the remaining capacity of bin j before packing the i-th group.

Algorithm 5 GreedySelectPack2

for $i \in [1..h], j \in [1..r]$ **do**
  Sort the groups in S_{ij} in increasing order of item-size.
  Add groups to $Selected_{ij}$ until the cumulative profit is in the range $[w_{ij}(\varepsilon^2\mathcal{O}/h), (w_{ij}+1)(\varepsilon^2\mathcal{O}/h)]$
end for
$Groups' = \bigcup_{i\in[1..h],j\in[1..r]} Selected_{ij}$
Sort $Groups'$ in decreasing order of item-size. Ties are broken arbitrarily.
for $G_i \in Groups'$ **do**
  Assign the items of G_i to the k_i bins with the highest γ_j^i values.
end for

The correctness of our PTAS relies on the following theorem.

Theorem 7. *If a feasible assignment of $Groups'$ exists, then the greedy algorithm finds one.*

Combining the above theorem with the analysis of the guessing-groups step, we have:

Corollary 6. AAP *with bounded-Cardinality, and items size that form a divisible sequence, admits a PTAS.*

D Hardness Results

D.1 Hardness of AAP

In this section we analyze the hardness of several restricted instances of AAP. Specifically, using a reduction from the 3-*partition* problem, we conclude that AAP is strongly NP-hard even for instances with unit-utilities, unit-group size, and no assignment restrictions. Next, we show that the presence of assignment restrictions makes the problem APX-Hard, even with unit-size, unit-utility and uniform-cardinality, using reduction from the 3- *bounded* 3-*dimensional matching problem* (3DM-3).

Theorem 8. AAP *is strongly NP-hard even for instances unit-utilities, unit-group size, and no assignment restrictions.*

Proof. We show a straightforward reduction from 3-*partition*, which is strongly NP-hard [5]. An instance of 3-partition is defined as follows.
Input: a finite multiset S of $n = 3m$ numbers, such that $\sum_{x\in S} x = mB$, and each $x \in S$ satisfies $B/4 < x < B/2$ for a given bound $B \in Z^+$.
Output: Is there a partition of S into m disjoint sets, $S_1, S_2, \ldots, S_m$, such that, for $1 \le i \le q, \sum_{x\in S_i} x = B$? (Note that the above constraints on the element sizes imply that every such S_i must contain exactly three elements from S).

Given an instance of 3-partition. For each number $x \in S$ there is group with a single item having size x and unit utility. There are no assignment restrictions and all m bins have capacity of B. It is easy to verify that an assignment with profit $3m$ exists if and only if a 3-partition exists. ■

The next hardness result implies that with the presence of assignment restrictions, an $\varepsilon_0 > 0$ exists such that it is NP-hard to decide whether an instance has an assignment with a profit p, or if every assignment has a profit of at most $(1 - \varepsilon_0)p$, i.e. there is no PTAS.

Theorem 9. AAP *is APX-hard even if all bins have unit-capacity, all groups have uniform-cardinality, items with unit-size, and unit-utility, and the assignment restrictions can match each bin to at most* 3 *groups.*

Proof. We use an approximation-preserving reduction from the maximum 3-bounded 3-dimensional matching (3DM-3) problem, defined as follows.
Input: A set of triplets $T \subseteq X \times Y \times Z$, where $|X| = |Y| = |Z| = n$; the number of occurrences of any item of $X \cup Y \cup Z$ in T is at most 3. The number of triplets is $|T| \geq n$.
Output: A 3-dimensional matching in T of maximal cardinality, i.e., a subset $T' \subseteq T$, such that any item in X, Y, Z appears at most once in T', and $|T'|$ is maximal.

Kann showed in [7] that 3DM-3 is APX-hard, i.e., $\varepsilon_0 > 0$ exists such that it is NP-hard to decide whether an instance has size n matching, or if every matching has a size of at most $(1 - \varepsilon_0)n$.

Given an instance I of 3DM-3-Matching, where $T \subseteq X \times Y \times Z$ and $X = \{x_1, ..x_n\}$, $Y = \{y_1, ...y_n\}$, $z = \{z_1....z_n\}$, construct the following instance I' of AAP. For each triplet in T define a group with cardinality 3, unit item-size, and unit utility. There are $3n$ bins $x_1, ..x_n, y_1, ...y_n, z_1....z_n$ each has unit-capacity. The assignment restriction is according to the triplets, i.e., items of the group (x_i, y_j, z_k) can be assigned only to the three bins x_i, y_j, z_k. A matching of size p, implies profit $3p$ for AAP. The matching consists of p disjoint triplets of T, each corresponding to a group whose items can be assigned to three different bins. For the other direction, note that a solution with profit $3p$ for AAP assigns p groups, such that the items (x_i, y_j, z_k) of each group, are assigned to the three bins x_i, y_j, z_k. The bins' unit-capacity immediately implies that the corresponding triplets form a matching of size p. ■

This hardness result implies that we can not have a PTAS for AAP with assignment restrictions.

D.2 Hardness of AGAP

In this section, we give hardness results for AGAP. We first show that AGAP is APX-hard already for very restricted instances.

Theorem 10. *Unless $P = NP$,* AGAP *cannot be approximated within factor $1 - 1/e$, even if all items have unit sizes, all bins have unit capacities, and the utility from packing an item ℓ in bin j is $a_{\ell j} \in \{0, 1\}$.*

Proof. We give a reduction from the *maximum coverage* problem. Given an instance of maximum coverage, let U be the ground set of elements, and let $S_1, \ldots, S_n \subseteq U$ be a collection of subsets of elements in U. We need to select k subsets $S_{i1}, \ldots, S_{ik}$ such that the number of elements covered by $\cup_{r=1}^{k} S_{ir}$ is maximized.

We use the following reduction. Define a bin of unit capacity for each element $j \in U$. Also, define a group of items, i, for each subset S_i, $1 \leq i \leq n$. The utility of assigning item ℓ in group i to bin j is $a_{\ell j} = 1$ if j is the element in S_i that corresponds to ℓ in group i, and for all other bins $a_{\ell j} = 0$. We add dummy bins and dummy items, so that we can fully satisfy the groups.

Consider a subset of groups, $S_{\pi 1}, \ldots S_{\pi k}$. We use the next claim.

Claim 12 *The maximum utility from packing $S_{\pi 1}, \ldots S_{\pi k}$ is equal to $|\cup_{r=1}^{k} S_{\pi r}|$.*

Proof. We show inequality in both directions. Clearly, the maximum profit from $S_{\pi 1}, \ldots S_{\pi k}$ is at most $|\cup_{r=1}^{k} S_{\pi r}|$, since we can get utility at most 1 for any item in their union.

We now show that the maximum profit $S_{\pi 1}, \ldots S_{\pi k}$ is at least $|\cup_{r=1}^{k} S_{\pi r}|$. This follows from the fact that we can first assign items in $S_{\pi 1}$ to bins with utility 1, then we can assign with utility 1 items in $S_{\pi 2} \setminus S_{\pi 1}$ and so on. ■

By Claim 12, there is a solution for maximum coverage which covers w elements iff there is a collection of groups in the AGAP instance whose packing yields utility w. ■

We now consider some natural variants of AGAP and show that all of them are hard to approximate. Consider first AGAP with no assignment restrictions on items of the same group.

Theorem 11. *If items of the same group can be packed in the same bin, then* AGAP *is hard to approximate within any bounded factor, already in the case of a single group and two bins.*

Proof. We show a straightforward reduction from PARTITION. Given an instance $a_1, ..., a_n$ for PARTITION, one can generate an instance for AGAP with no assignment restrictions on items of the same group. Consider an instance where there are only two bins and one group G, with utility 1, which contains n items of sizes $s_i = \frac{2a_i}{\sum_{j=1}^{n} a_j}$. Note that we can pack G iff there is partition of $\{a_i\}$ into two sets. In this case, we get a total utility of 1; otherwise, the utility is 0. A polynomial time approximation algorithm for AGAP would enable to distinguish between the two cases, thus solving PARTITION in polynomial time. ■

Consider now AGAP instances in which some bins are *forbidden* for some of the items. We show that with no restriction on F, the maximum number of

forbidden bins for an item, the problem cannot be approximated within any bounded ratio, even if the profit of the item does not depend on the bin in which it is packed.

Theorem 12. AGAP *with forbidden bins is hard for any bounded approximation, even if the profit of each item is uniform across all bins.*

Proof. By reduction from *packing integer program (PIP)*. Given $A \in \{0,1\}^{m\times n}$, $b \in \mathbb{N}$ and $p \in [0,1]^n$ a PIP seeks to maximize $p^T x$ subject to $x \in \{0,1\}^n$ and $Ax \leq b$. We show how to solve an instance of PIP via a reduction to AGAP with forbidden bins. Consider the following groups $G_1, ..., G_n$, where G_i consists of the items $a_{i,j}$, for $j \in [m]$. Each item $a_{i,j}$ can be only in bin j (all the other bins are forbidden for $a_{i,j}$). The size of $a_{i,j}$ is $A_{i,j}/b_j$. Let the profit of G_i be p_i. We note that each group can be packed in a unique way, therefore a solution is determined only by the set $S \subseteq [n]$. One can easily verify that a set S is a feasible solution of this AGAP instance iff x, its indicator vector, is a feasible solution for the PIP instance. Since we used for the reduction an AGAP instance in which item profits are identical across the bins, and the total profit is $\sum_{i\in S} p_i$, we get that AGAP with forbidden bins is harder for approximation than PIP. ■

Consider now a generalization of AGAP in which item utilities may be *negative*. We note that this version of AGAP is harder than AGAP with forbidden bins. Indeed, we can simulate a forbidden bin for an item by setting the item utility in this bin to be infinitely negative. Thus, we have the following.

Corollary 7. AGAP *with negative utilities is hard to approximate within any bounded ratio.*

We now turn to AGAP instances with arbitrary bin capacities.

Theorem 13. *AGAP with non-identical bins is hard for any constant approximation, even if the profit of an item is identical across the bins.*

Proof. As in the previous result, we use a reduction from PIP. Given an instance $A \in \{0,1\}^{m\times n}$, $b \in \mathbb{N}$ and $p \in [0,1]^n$, we construct the following AGAP instance. Each bin j has capacity $\frac{1}{(3n)^{j-1}\prod_{k=1}^{j-1}\left(b_k+\frac{1}{2}\right)}$. The groups are $G_1, ..., G_n$, where G_i has profit p_i and consists of the items $a_{i,j}$, for $j \in [m]$. Let the size of an item $a_{i,j}$ be $s_{i,j} = \frac{A_{i,j}+\frac{1}{2n}}{(3n)^{j-1}\prod_{k=1}^{j}(b_k+\frac{1}{2})}$. We note that for $k > j$, $s_{i,j}$ is grater than the capacity of bin k. We conclude that item $a_{i,j}$ can be packed only in bin j: it cannot be packed into bin k for $k > j$, which is too small, and it cannot be packed into bin k for $k < j$, which must be occupied by an item $a_{i,j'}$, where $j' < j$. Let $S \subseteq [n]$ be a set of groups. S is a feasible solution iff for every bin j, $\sum_{i\in S} s_{i,j}$ is at most the capacity of bin j. This condition is equivalent to $\sum_{i\in S}(A_{i,j} + \frac{1}{2n}) \leq b_j + \frac{1}{2}$. Let x be the indicator vector of S. Because $A_{i,j} \in \{0,1\}$, S is a feasible solution iff, for all j, $\sum_{i=1}^{n} A_{i,j}x_i \leq b_j$, i.e. x is a feasible solution of the PIP instance. Note that the objective function of the two problems is the same ($\sum_{i=1}^{n} x_i p_i$). This yields the statement of the theorem. ■

E Some Proofs

Proof of Claim 1: It is easy to see that f is non-decreasing. Indeed, the sum of fractional knapsack values of the bins in $[m]$ can only increase when $|S|$ increases. We proceed to show that f is submodular. Given $S \subseteq U$, since $f(S)$ is the sum of fractional utilities of the bins, it is enough to show that the fractional utility $f_j(S)$ of any bin j is a submodular function on U. It is enough to show that for any subsets $S \subset T \subseteq U$, and $s \notin S$, we have

$$f_j(S \cup \{s\}) - f_j(S) \geq f_j(T \cup \{s\}) - f_j(T).$$

Now let $s = (G, L)$. If L has no element in G assigned to bin j, it is easy to see that $f_j(S \cup \{s\}) - f_j(S) = f_j(T \cup \{s\}) - f_j(T) = 0$. Therefore assume that L assigns a unique element $\ell \in G$ to bin j. Note that the value $f_j(W)$ for $W \subseteq U$ is computed by ordering the elements ℓ' assigned to bin j by assignments in W in the non-increasing order of $a_{\ell' j}/s_{\ell'}$ and taking the (fractional) profit from a (fractional) prefix with the sum of sizes summing up to 1. Now it is easy to see that the value $f_j(S \cup \{s\}) - f_j(S)$ (resp., $f_j(T \cup \{s\}) - f_j(T)$) depends on the position of ℓ in this order for S (resp. T). Since $S \subset T$, the position of ℓ in the order for S is no later than its position in the order for T. Therefore, we have $f_j(S \cup \{s\}) - f_j(S) \geq f_j(T \cup \{s\}) - f_j(T)$ as desired. ■

Proof of Lemma 2: For $h \in [H]$, let $y_h = \lceil (\sum_{(G,L) \in S: G \in \mathcal{C}_h} \sum_{\ell \in G} s_\ell)/(\varepsilon m/H) \rceil$. Since S satisfies Constraint 2, we have $\sum_{h=1}^{H} y_h \leq H/\varepsilon + H$. Thus the vector $(y_1, \ldots, y_H)$ is legal. From the definition of y_h and the fact that any group in $\mathcal{C}_h$ has size at least $\varepsilon m/n \cdot (1+\varepsilon)^{h-1}$, the lemma follows. ■

Proof of Lemma 3: Note that Constraint 1 (resp. Constraint $2'$) alone defines a partition matroid. Since the partition of Constraint 1 is a refinement of the partition of Constraint $2'$, they together form a laminar matroid. Now any item in class h has size at most $\varepsilon m/n \cdot (1+\varepsilon)^h$. This together with the definition of Constraint $2'$ and that of a legal vector, implies that the total size of groups in $G(S)$ in classes 1 to H is at most $m(1+\varepsilon)^2$. Class 0 contributes at most εm total size. The lemma thus follows. ■

Proof of Lemma 5: In the following we prove that AGAP-LP can be solved efficiently. This is done by proving that the dual of AGAP-LP has an efficient separation oracle Below is the dual of AGAP-LP.

$$\begin{aligned}
\text{(Dual) } \min \quad & \sum_{i \in [n]} z_i + \sum_{j \in [m]} c_j y_j + M \cdot W & & \\
\text{s.t.} \quad & z_i + \sum_{j \in [m] \mid \exists\, \ell \in G_i : p(\ell) = j} s_\ell y_j + W \cdot \sum_{\ell \in G_i} s_\ell \geq \sum_{\ell \in G_i} a_{\ell p(\ell)} & & \forall\, i \in [n], p \in \mathcal{P}_i \\
& z_i \geq 0 & & \forall\, i \in [n] \\
& y_j \geq 0 & & \forall\, j \in [m] \\
& W \geq 0 & &
\end{aligned}$$

Given a non-negative solution for the dual LP, it is infeasible if there exists a group i for which there is an admissible packing $p \in \mathcal{P}_i$ such that:

$$z_i < \sum_{\ell \in G_i} a_{\ell p(\ell)} - \sum_{j \in [m] | \exists\, \ell \in G_i : p(\ell) = j} s_\ell y_j \quad - W \cdot \sum_{\ell \in G_i} s_\ell = \sum_{\ell \in G_i} [a_{\ell p(\ell)} - s_\ell (y_{p(\ell)} + W)] \ .$$

Given group i, deciding whether there exists an admissible packing $p \in \mathcal{P}_i$ such that the above inequality holds can be reduced to matching the following way. Construct a bipartite graph whose left side is the set of elements in G_i and its right side is the set of bins. For every element $\ell \in G_i$ and bin j, the edge between their corresponding nodes has a weight of $a_{\ell j} - s_\ell(y_j + W)$. It is clear that any matching of size $|G_i|$ corresponds to a admissible packing p of G_i such that the right hand side of the above inequality for i and p is equal to weight of the matching, and vice versa. Hence, one can find the admissible packing maximizing the right hand side of the above inequality using a maximum weight matching algorithm. ■

Proof of Theorem 3: We show that AGAP with unit item sizes can be cast as a special case of maximizing a submodular function subject to a knapsack constraint. Let $\mathcal{G} = \{G_1, \ldots, G_p\}$ be a collection of groups. Given $S \subseteq \mathcal{G}$, define

$$f(S) = \{\text{maximum profit from packing items } d \in S \text{ subject to bin capacities}\}.$$

It is easy to verify that f is monotone and submodular. Thus, AGAP with unit sized items can be formulated as follows.

$$\max_{S \in \mathcal{G}} \quad f(S)$$

$$\textit{subject to}: \quad \sum_{i \in S} k_i \leq m$$

Using a result of [20] we get a $(1 - 1/e)$-approximation for the problem. ■