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

Ron Adany† Moran Feldman‡ Elad Haramaty‡ Rohit Khandekar§
Baruch Schieber¶ Roy Schwartz‖ Hadas Shachnai‡** Tami Tamir††

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 a 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 recent media research reports [20, 19], global spending on TV ads exceeded $323B in 2011, and an average viewer watched TV over 153 hours per month, with the average viewing time 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

*A preliminary version of this paper appeared in the Proceedings of the 16th Conference on Integer Programming and Combinatorial Optimization (IPCO), Valparaíso, March 2013.

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

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

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

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

‖Microsoft Research, One Microsoft Way, Redmond, WA 98052. E-mail: roysch@microsoft.com

**Work partially supported by the Technion V.P.R. Fund, by Smoler Research Fund, and by funding for DIMACS visitors.

††School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. E-mail: tami@idc.ac.il.

and for mobile operators [6, 12, 23]. 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 [11]. Our study is motivated by a central application in personalized ad campaigns scheduling, introduced to us by SintecMedia [25] and fully described in [1].

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.[1] 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.

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

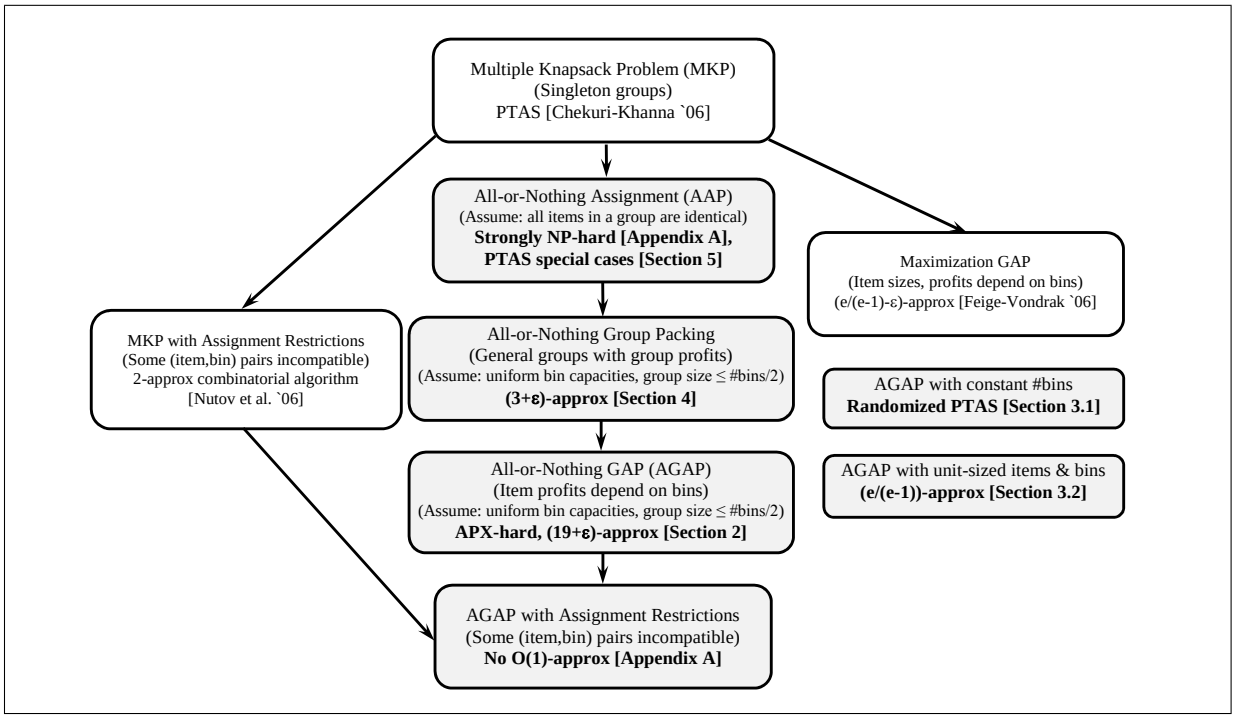

Figure 1: Summary of our approximation and hardness results and comparison with related problems. An arrow from problem A to B indicates that A is a special case of B.

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 A). 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 the solution output by $\mathcal{A}$ and by 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 A.5). 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, with unit bin capacities, an $\frac{e}{e-1}$-approximation can be obtained by reduction to submodular maximization with a knapsack constraint. In Section 4 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 Section 5 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,[2] 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 [28, 18]. Also, there are standard sizes of $150, 300$ and 600 pixels for web-banners on the Internet [27].

Finally, we present (in the Appendix) 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 a 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 without actually applying this knapsack constraint. Instead, we encode the knapsack constraint as a matroid constraint, by considering feasible vectors of logarithmic size that represent the number of groups taken from each size category. These vectors (which are *implicitly* enumerated in polynomial time) are used for defining the matroid constraint.

Our definition of the submodular set function, $f(S)$, on the way to finding a *fractional* packing of items (see Section 2), in fact guarantees that the rounding that we use for group sizes causes only small harm to the approximation ratio. This allows also to define a non-standard polynomial time implementation of an algorithm of Calinescu et al. [3], for maximizing a submodular function under a 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 [13, 14] for maximizing a submodular function subject to a fixed number of knapsack constraints.

1.2 Related Work

All-or-nothing GAP, and its more restricted variants AAP, generalize several classical problems, including GAP (with same sizes across the bins), *multiple knapsack* (MKP), and *multiple knapsack with assignment restrictions* (MKAR) [21].

Recall that the special case of AGAP 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 [5]. Fleischer et al. [9] obtained an $\frac{e}{e-1}$-approximation for GAP, as a special case of the *separable assignment problem*. The best known ratio is $\frac{e}{e-1} - \varepsilon$ [8].

In minimum GAP (see, e.g., [15]), 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

[2] A sequence $d_1 < d_2 < \cdots < d_D$ is *divisible* if d_{i-1} divides d_i for all $1 < i \leq D$.

to be assigned to a single machine. The paper [24] 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 total costs, and all the jobs must be assigned. The only differences from GAP are the multiple assignment constraints of each job. The paper [22] presents Lagrangian dual-based branch-and-bound algorithms that yield exact solutions for the problem.[3]

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 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

2.1.1 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. We note that since the total size of the items assigned to a single bin may exceed its unit capacity, the profit from an (integral) assignment of groups to the bins may be smaller that the total utility of these items. Thus, we define the utility contribution of a bin $j \in [m]$ as the maximum value

[3]The running times of these algorithms are not guaranteed to be polynomial.

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 2.1 *Given a subset $I \subseteq [N]$ of items and a bin j, define $\pi(j, I) = \max_{\vec{w}} \sum_{\ell \in I} w_\ell a_{\ell j}$, where the maximum is taken over all weight vectors $\vec{w} \in \Re_+^{|I|}$ 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 $\vec{w}$ that maximizes the utility contribution of bin j. Order the items in I as $\ell_1, \ldots, \ell_b$ in a non-increasing order of their ratios 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 $w_{d+1} = (1 - s)/s_{\ell_{d+1}}$ and set the other weights $w_{d+2} = \cdots = w_b = 0$. If $s = 1$, set weights $w_{d+1} = \cdots = w_b = 0$.

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 2.1 to *multiset* I of $[N]$ as follows.

Definition 2.2 *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 $\pi(j, I) = \max_{\vec{w}} \sum_{\ell \in [N]} w_\ell a_{\ell j}$, where the maximum is taken over all weight vectors $\vec{w} \in \Re_+^N$ that assign weights $w_\ell \in [0, I(\ell)]$ to $\ell \in [N]$ satisfying $\sum_{\ell \in [N]} w_\ell s_\ell \leq 1$.*

2.1.2 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

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

Claim 2.1 *The function $f(S)$ is non-decreasing and submodular.*

Proof: 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. ∎

To identify subsets $S \subseteq U$ 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 $\varepsilon > 0$. 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 $\binom{H+H(1+1/\varepsilon)}{H} = O(e^{(2+1/\varepsilon)(H+1)})$, which is polynomial in n.[4]

Lemma 2.1 *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 $\hat{y}_h := \lfloor y_h n/(H(1+\varepsilon)^{h-1}) \rfloor$.*

Proof: 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. ∎

This lemma implies, in particular, that the optimum solution to AGAP satisfies the above property as well. With this motivation, we denote by $U_h = \{(G, L) \in U \mid G \in \mathcal{C}_h\}$ the collection of groups 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 groups in $G(S)$, counted with multiplicities, that are in $\mathcal{C}_h$ is at most $\hat{y}_h$, where $\hat{y}_h$ is defined in Lemma 2.1.

Lemma 2.2 *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 $\sum_{(G,L)\in S} \sum_{\ell\in G} s_\ell \leq m((1+\varepsilon)^2 + \varepsilon)$.*

[4]The last equation follows from a result of [5].

Proof: 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 group 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. ∎

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. [17] proved that a greedy algorithm that starts with an empty set and iteratively adds a "most profitable" element to it while maintaining independence, as long as possible, is a 2-approximation. Each iteration can be implemented in polynomial time as follows. Given the 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 the 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 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 $e/(e-1)$. Instead of the greedy algorithm of Nemhauser et al. [17], we can also use the $\frac{e}{e-1}$-approximate Continuous Greedy Algorithm of Calinescu et al. [3]. This algorithm starts with an empty solution $S = \emptyset$, and improves it in rounds. In each round it finds an independent set maximizing a linear objective. The weight of each element (G, L) in the linear objective is the marginal value of (G, L) with respect to the original objective function f, given the current fractional solution. The algorithm then updates its current solution, by making a small step in the direction of this independent set. The fact that the size of the ground set is exponential prevents a straight-forward implementation of the algorithm for two reasons:

- Calculation of the weights of all ground set elements will require exponential time.
- The number of iterations required is polynomial in the size of the ground set, which in turn is exponential in the size of the input.

The first difficulty can be eliminated by observing that any independent set can contain at most one assignment for each group; therefore, it suffices to consider only the 'best' assignment for each group, given the current fractional solution. A technique for finding such an assignment (and its weight) using the bipartite matching algorithm was described earlier, in the discussion of the greedy algorithm. The second difficulty can eliminated by observing that, with slight changes, the proof of Calinescu et al. is valid also when the number of rounds is reduced to r^2, where r is the rank of the matroid, which in turn is at most n, the number of groups. Note also that the total number of non-zero entries in the final fractional solution computed is at most the number of groups, n, multiplied by the number of rounds, at most n^2, and hence is at most n^3.

The output of the Continuous Greedy Algorithm is a fractional value corresponding to a point inside the matroid polytope. Rounding without any loss can be done, e.g., by the the pipage rounding technique of [2]. The standard analysis of pipage rounding shows that it runs in time polynomial in the size of the ground set. However, an easy observation shows that elements with zero values in the fractional solution are never accessed by the rounding algorithm. Thus, it in fact runs in time polynomial in the number of non-zero entries in the input. Since the input to the pipage rounding is the output of the Continuous Greedy Algorithm, which has at most n^3 non-zero entries, the pipage rounding step can be applied in polynomial time as well.

In summary, we find a set $S^* \subseteq U$ such that

(i) each group appears at most once in $G(S^*)$,

(ii) 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

(iii) $f(S^*)$ is at least $(e-1)/e$ times 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 values will be optimized later.

Lemma 2.3 *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 $\vec{w}(j)$. Fix any such optimum weight vector $\vec{w}^*(j)$ for each j. These vectors, when combined, give a weight vector $\vec{w}^* \in \Re_+^N$, 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 $(G, L) \in S^*$.

2.2.1 Proof of Lemma 2.3

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 Section 2.1) 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 claim that $f(T) \geq f(S^*) \cdot (1/(v(1+4\varepsilon)) - 1/u)$. If the size of the groups in T is exactly m/v, then since the density of T is at least the overall density of $G(S^*)$, we have

$f(T) \geq f(S^*)/(m(1+4\varepsilon))\cdot(m/v) = f(S^*)/(v(1+4\varepsilon))$. Otherwise, consider the group G following T in the sequence of the groups in $G(S^*)$ ordered in non-increasing order of density. Since the size of the groups in $T \cup \{G\}$ is greater than m/v, we have $f(T \cup \{G\}) > f(S^*)/(m(1+4\varepsilon))\cdot(m/v) = f(S^*)/(v(1+4\varepsilon))$. The claim follows since $\sigma^*(G) < f(S^*)/u$.

The three steps below find a feasible solution to AGAP that consists of groups in $G(T)$ and whose value is at least $f(T)/2$.

1. Eliminate all zero weights Let $\vec{w} \in \Re_+^N$ 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. Recall that all items in this bin but one have weight 1. We keep in this bin either all the items assigned to it that have weight 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 there are at most a such items from full bins, and at most b such items from half full bins. These items can be packed into evicted $\lfloor (m-a)/2 \rfloor$ bins if $a + b \leq \lfloor (m-a)/2 \rfloor$. This is indeed true since $v \geq 4$ together with $a + b/2 \leq m/v$ implies $4a + 2b \leq m$, which in turn implies $a + b \leq (m-a)/2$. It follows that $a + b \leq \lfloor (m-a)/2 \rfloor$, since $a + b$ is an integer.

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, and 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., $k_i \leq \lceil m(1-2/v) \rceil$.

Recall that $f(T) \geq f(S^*)\cdot(1/(v(1+4\varepsilon)) - 1/u)$. 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. ∎

Now, to bound the overall approximation ratio, we set $1/u = \frac{1}{2}(1/(v(1+4\varepsilon)) - 1/u)$, i.e., $u = 3v(1+4\varepsilon)$. Thus, with $v = 4$ and $u = 12(1+4\varepsilon)$, we get a ratio of $\frac{1}{12(1+4\varepsilon)}$. 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 $12(1+4\varepsilon)\frac{e}{e-1}$-approximation).

This proves the following theorem.

Theorem 2.4 AGAP *with uniform bin capacities admits a polynomial-time* $12(1+\varepsilon)\frac{e}{e-1}$*-approximation for any* $0 < \varepsilon < 1$ *provided any group has at most* $k_{\max} \leq m/2$ *items.*

3 Approximating Special Cases of AGAP

In this section we consider two special cases of AGAP: In Section 3.1, we give a randomized PTAS for instances with fixed number of bins. In Section 3.2 we present an $\frac{e}{e-1}$-approximation for instances where items have the same (unit) size, and all bins have unit capacity.

3.1 LP-Based Approximation Scheme for Fixed Number of Bins

We formulate the following LP relaxation for AGAP. For every group G_i, let $\mathcal{P}_i$ be the collection of admissible packings of elements of group G_i alone (i.e., $p(j) = \perp$ for every packing $p \in \mathcal{P}_i$ and element $j \notin G_i$). The relaxation has an indicator variable $x_{i,p}$ for every group G_i and admissible assignment $p \in \mathcal{P}_i$.

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

Constraint (1) requires every group to have at most one assignment. Constraint (2) guarantees that no bin is over-packed. In this section we will be interested in the case where the number of bins is a constant. Notice that in this case AGAP-LP has a polynomial size, and thus, can be solved in polynomial time using standard techniques. However, for completeness, we point out that AGAP-LP can be solved in polynomial time even for non-constant number of bins.

Lemma 3.1 *AGAP-LP can be solved in polynomial time for non-constant number of bins.*

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

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

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:

$$z_i < \sum_{\ell \in G_i} a_{\ell p(\ell)} - \sum_{j \in [m] | \exists \ell \in G_i : p(\ell) = j} s_\ell y_j = \sum_{\ell \in G_i} [a_{\ell p(\ell)} - s_\ell y_{p(\ell)}] \ . \tag{1}$$

Given group i, deciding whether there exists an admissible packing $p \in \mathcal{P}_i$ such that (1) holds can be reduced to a matching problem in 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 such that $s_\ell \leq c_j$, the edge between their corresponding nodes has a weight of $a_{\ell j} - s_\ell y_j$. It is clear that any matching M of size $|G_i|$ corresponds to an admissible packing p of G_i inducing a value equal to the weight of M on the right hand side of (1), and vice versa. Hence, one can find the admissible packing maximizing the right hand side of (1) using a "maximum weight matching of size k" algorithm. ∎

The next theorem presents an approximation scheme for constant number of bins. This scheme draws many ideas from the rounding procedure suggested in [13, 14] for the problem of maximizing a submodular function subject to a constant number of knapsack constraints.

Theorem 3.2 *There is a randomized polynomial time approximation scheme for* AGAP *with fixed number of bins.*

The rest of this subsection is devoted to proving Theorem 3.2. 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 $\ell \in [N]$:

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

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). Algorithm 1 begins by guessing the set $\mathcal{A}$ of the most valuable groups of OPT and their corresponding assignment in OPT. Then, the algorithm dismisses all assignments of groups outside of $\mathcal{A}$ assigning an element ℓ to a bin j such that s_ℓ is large in comparison to the remaining capacity r_j of j. Finally, the assignment is completed by a randomized rounding of an AGAP-LP solution.

Lemma 3.3 *For a constant ε, Algorithm 1 has a polynomial time complexity.*

Proof: It is clear that all parts of the algorithm beside Lines 1 and 2 can be performed in polynomial time. Let us show that these lines can also be implemented 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:

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

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 group $G_i \in \mathcal{A}$ the packing p'_i induced by OPT.
Remove all groups of $\mathcal{A}$ from AGAP-LP.
for every bin $j \in [m]$ **do**
    Set 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 $s_\ell \geq \varepsilon^3 m^{-2} \cdot r_{p(\ell)}$.
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 P is an admissible packing **then**
    Return P.
else
    Return an empty packing (i.e., a packing assigning all elements to $\perp$).
end if

Hence, the number of possible pairs is only:

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

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

Let us now consider the approximation ratio of Algorithm 1. The algorithm dismisses large assignments of groups which does not belong to $\mathcal{A}$. 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 3.4 *The total revenue of the packings in $\mathcal{P}_B$ is at most $\varepsilon \cdot OPT$. Formally, $\sum_{p'_i \in \mathcal{P}_B} \sum_{\ell \in G_i} a_{\ell p'_i(\ell)} \leq \varepsilon \cdot OPT$.*

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 $j \in [m]$, the union bound gives:

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

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

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

We can now lower bound the value of the solution y of AGAP-LP that the algorithm calculates. 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 3.5 *The solution y of AGAP-LP produced by Line 9 of Algorithm 1 has value of at least $(1-\varepsilon)OPT - R_A$.*

Proof: Let us construct a packing OPT' inducing a feasible (integral) solution y. 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 if OPT' packs a group $G_i \notin \mathcal{A}$, then it does so with a packing p for which the variable $x_{i,p}$ is not removed by Algorithm 1. Moreover, the total size of the elements packed by OPT' into bin j is at most:

$$\sum_{\ell \in [N] | OPT'(\ell) = j} s_\ell \quad \leq \sum_{\ell \in [N] | OPT(\ell) = j} s_\ell \quad - \sum_{p \in \mathcal{P}_A | \exists\, \ell : p(\ell) = j} s_\ell \quad \leq c_j - \sum_{p \in \mathcal{P}_A | \exists\, \ell : p(\ell) = j} s_\ell \quad = r_j \ .$$

Thus, OPT' induces a feasible solution for the AGAP-LP instance that algorithm solves on Line 9. The corollary now follows since, by Lemma 3.4, OPT' has a revenue of at least $(1-\varepsilon)OPT - R_A$. ∎

Recall that P is a (possibly infeasible) packing constructed by the algorithm. Let R_P denote the total revenue of P. The following corollary lower bounds the expectation of R_P.

Corollary 3.6 $\mathbb{E}[R_P] \geq (1-2\varepsilon)OPT$.

Proof: Every packing p of a group $G_i \notin \mathcal{A}$ gets merged 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:

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

which is exactly $1-\varepsilon$ times the value of y. By Corollary 3.5, the value of y is at least $(1-\varepsilon)OPT - R_A$. Hence, the groups of $[n] \setminus \mathcal{A}$ contribute to R_P, in expectation, at least $(1-\varepsilon) \cdot [(1-\varepsilon)OPT - R_A] = (1-\varepsilon)^2 OPT - (1-\varepsilon)R_A \geq (1-2\varepsilon)OPT - R_A$.

Notice that the contribution of the groups of $\mathcal{A}$ to R_P is exactly R_A. The lemma now follows by adding up 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 3.6. 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 $E_{\mathcal{A}}$ be the set of elements in the groups of $\mathcal{A}$; formally, $E_{\mathcal{A}} = \cup_{G_i \in \mathcal{A}} G_i$. Let Q_j be the ratio $r_j^{-1} \cdot \sum_{\ell \in [N] \setminus E_{\mathcal{A}} | P(\ell)=j} s_\ell$, and let $Q = \max_{j \in [m]} Q_j$. Observe that the algorithm outputs P if and only if $Q \leq 1$.

Lemma 3.7 *For every bin $j \in [m]$ and $h \geq 1$, $\Pr[Q_j \geq h] \leq \varepsilon m^{-2} h^{-2}$.*

Proof: After the initialization $P = \odot_{G_i \in \mathcal{A}} p'_i$, the remaining capacity of bin j is r_j. Let L_j be the total size of the elements of groups outside of $\mathcal{A}$ that are added to bin j in P after the initialization. Formally,

$$L_j = \sum_{i \in [n] | G_i \notin \mathcal{A}} \sum_{\ell \in G_i | P(\ell)=j} s_\ell \ .$$

It is enough to upper bound the probability that L_j exceeds hr_j. Let E_j be the set of elements that do not belong to a group of $\mathcal{A}$ and have a positive probability to be packed by P into bin j. For every element $\ell \in E_j$, let $q_\ell > 0$ be the probability of ℓ to be packed by P into bin j, and let 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 $\Pr[\sum_{\ell \in E_j} s_\ell X_\ell \geq hr_j] \leq \varepsilon m^{-2} h^{-2}$. Let us state some observations applying to the elements of E_j.

- The probability of each element of E_j to be packed in bin j is determined by the solution y of AGAP-LP, hence, $\sum_{\ell \in E_j} s_\ell q_\ell \leq (1-\varepsilon) r_j$.
- Large packings are removed by Algorithm 1, thus, $s_\ell \leq \varepsilon^3 m^{-2} r_j$ for every $\ell \in E_j$.
- 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., $X_{\ell_1} \cdot X_{\ell_2} = 0$).

The first observation implies that the expected value of the sum $\sum_{\ell \in E_j} s_\ell X_\ell$ is at most $(1-\varepsilon) r_j$.

Let us use the other three observations to lower bound the variance of this sum.

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

We are now ready to bound $\Pr[\sum_{\ell\in E_j} s_\ell X_\ell > h r_i]$ using Chebyshev's inequality.

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

■

Corollary 3.8 *For every $h \ge 1$, $\Pr[Q \ge h] \le \varepsilon m^{-1} h^{-2}$.*

Proof: Follows from Lemma 3.7 and the union bound. ■

Lemma 3.9 *If $Q \le h$ for some $h > 1$, then $R_P \le 4mh \cdot OPT$.*

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 forms 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, each time two prefixes are removed, the total size of the elements packed into some 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 Theorem 3.2.

Proof: [Proof of Theorem 3.2] The expected revenue of Algorithm 1 is $\mathbb{E}[R_P|Q \le 1]$. Let us lower bound this quantity as following:

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

By Corollary 3.6, $\mathbb{E}[R_P] \geq (1-2\varepsilon)OPT$. By Corollary 3.8, $\Pr[2^{h-1} < Q \leq 2^h] \leq \Pr[2^{h-1} \leq Q] \leq \varepsilon m^{-1} 2^{2-2h}$. Finally, by Lemma 3.9, $\mathbb{E}[R_P | 2^{h-1} < Q \leq 2^h] \leq 4m \cdot 2^h \cdot OPT$. Combining all these observations into (2) gives:

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

■

3.2 Approximation Algorithm for Unit Size Items

We now discuss a special case of AGAP in which all items have unit sizes, and all bins have unit capacity. For such instances we obtain the best possible approximation ratio.

Theorem 3.10 AGAP *with unit item sizes and unit bin capacities admits an* $\frac{e}{e-1}$*-approximation.*

Proof: 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.

$$\begin{aligned} \max_{S \in \mathcal{G}} \quad & f(S) \\ \textit{subject to}: \quad & \sum_{i \in S} k_i \leq m \end{aligned}$$

Using a result of [26], we get an $(e/(e-1))$-approximation for the problem. ■

4 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, $1 \leq i \leq n$. We assume throughout the discussion that $k_{\max} \leq m/2$.

We give below a $(3+\varepsilon)$-approximation algorithm for the problem. Our bound (in Theorem 4.1), 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 $\max\{U_1, U_2\}$.

Theorem 4.1 *Algorithm GroupPack yields a* $(\frac{2(\gamma+1)}{\gamma}+\varepsilon)$*-approximation for all-or-nothing group packing, with uniform bin capacities and groups of cardinalities at most* $k_{max} \leq m/2$.

We use in the proof the next lemmas.

Lemma 4.2 *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 4.3 *Given an input for all-or-nothing 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* $[OPT(\frac{1}{2} - \frac{1}{\gamma c}), \frac{OPT}{2})$*, 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 into 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 $L_2, L_3, \ldots$.

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 groups 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 (*i*) does not hold, and $c \geq 2$, L_r consists of at least γ groups. Recall that the groups in L_r are added in non-increasing order by utilities. Consider the first γ groups in L_r, then each of this groups has total utility $\frac{OPT}{\gamma c}$ 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 4.1: 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 $\alpha m = k_{max}$ bins, assigning a single item to each bin. Otherwise, we select the set of groups output by the FPTAS, in Step 1. By Lemma 4.2, 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 4.3, there exists a subset of groups of total utility at least $\min\{\frac{OPT}{c}, OPT(\frac{1}{2} - \frac{1}{\gamma c})\}$. Such a subset is output by the algorithm. Indeed, GroupPack either find (in Step 1) a subset satisfying condition (ii) of the lemma, or (in Step 2) a subset satisfying condition (i). The approximation ratio is obtained by taking $c = \frac{2(\gamma+1)}{\gamma}$. ■

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 4.4 *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 $k_{max} \leq \frac{m}{2}$.*

5 Approximation Schemes for Subclasses of AAP

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

Recall that in AAP, the items form n groups, where group G_i, for $i \in [n]$, consists of k_i items, each of size s_i and utility a_i, independent of the bin to which the item is assigned. Denote by $P(G_i)$ the *group utility* of group G_i, i.e., $P(G_i) = k_i \cdot a_i$, which is the utility obtained from packing G_i. For a set of groups S, let $P(S)$ be $\sum_{G \in S} P(G)$. Let $P_{\max} = \max_{i \in [n]} P(G_i)$. Our approximation schemes consist of a preprocessing step, in which item utilities are scaled and rounded; then, the set of groups to be assigned is selected and ordered, and finally packed greedily. The group selection and ordering step is implemented differently, depending on our assumptions on the instance. We show how to implement this step for instances with unit-size items, and for instances where the item sizes form a divisible sequence, and the number of different group cardinalities is a fixed constant.

Algorithm 3 Overview of the PTAS

1: Guess the approximate overall utility $\mathcal{O}$, such that $\max\{P_{\max}, (1-\varepsilon)OPT\} \leq \mathcal{O} \leq OPT$.
2: Discard all groups i for which $P(G_i) \leq \varepsilon\mathcal{O}/n$.
3: Scale all group utilities by $\varepsilon/n \cdot \mathcal{O}$, such that after scaling $P(G_i) \in [1, n/\varepsilon]$, for $i \in [n]$.
4: Round down $P(G_i)$ to the nearest power of $(1+\varepsilon)$, for all $i \in [n]$.
5: Partition the groups into $h \leq (\alpha/\varepsilon)\ln n$ distinct sets $S_1, \ldots, S_h$, for some constant α, such that each set consists of groups with identical scaled and rounded group utility.
6: Guess the approximate (scaled and rounded) overall utility from each utility category S_q in some optimal solution U. Specifically, guess a tuple $(w_1, \ldots, w_h)$ such that $w_q \in [h/\varepsilon^2]$, and $w_q(\varepsilon n/h) \leq P(U \cap S_q) \leq (w_q+1)(\varepsilon n/h)$.
7: Select the groups to be packed and order them (depending on the instance).
8: Pack the selected groups greedily using the sorted list.

Whenever 'guessing' is used, the algorithm actually performs a search over a polynomial-size range, similar to the PTAS for MKP [5]. First, we guess the approximate value $\mathcal{O}$ of some

optimal solution. Since $P_{\max} \leq \mathcal{O} \leq n \cdot P_{\max}$, the guessing of $\mathcal{O}$ can be done using a binary search over the values in $\{P_{\max} \cdot (1+\varepsilon)^{\ell} | \ell \in [(1/\varepsilon)\ln n]\}$. In the other guessing step (line 6), we guess the approximate utility from each utility category in some optimal solution, while ignoring those categories that contribute less than $\varepsilon n/h$ to the scaled and rounded utility, as the total contribution of these sets is negligible. Since the total utility is $\leq n/\varepsilon$, and since we ignore groups of negligible size, $\sum_q w_q \leq \lceil h/\varepsilon^2 \rceil$. The number of h-tuples $(w_1, \ldots, w_h)$, such that $w_q \in [0, h/\varepsilon^2]$ and $\sum_q w_q \leq \lceil h/\varepsilon^2 \rceil$, is $O(n^{O(1/\varepsilon^3)})$ (follows from Stirling's approximation, see also Claims 2.4 and 2.5 in [5]). We get that the overall guessing process requires $O(\ln n) \cdot O(n^{O(1/\varepsilon^3)})$ time.

We claim that the revenue loss due to the instance transformation is bounded by a factor of $O(\varepsilon)$. Discarding groups of utility $P(G_i) \leq \varepsilon\mathcal{O}/n$ (line 2) may cause a loss of at most $n \cdot \varepsilon\mathcal{O}/n = O(\varepsilon)OPT$. Rounding down the group utilities (as well as the value of $\mathcal{O}$), to the nearest power of $(1+\varepsilon)$ may cause a loss of at most factor of $\varepsilon/(1+\varepsilon)$. Finally, guessing the h-tuple may cause a loss of at most εn to the scaled and rounded solution.

Let $G_{\pi(1)}, \ldots, G_{\pi(\ell)}$ be the ordered set of groups selected in line 7 of Algorithm 3. We describe the greedy algorithm used to pack these groups. Denote by $(\gamma_1^i, \ldots, \gamma_m^i)$ the remaining capacity of the bins before packing the items of group $G_{\pi(i)}$. Initially, for all bins $j \in [m]$, $\gamma_j^1 = c_j$. For $i \in [\ell]$, the $k_{\pi(i)}$ elements of $G_{\pi(i)}$ are assigned to the $k_{\pi(i)}$ bins with the highest γ_j^i values. Ties are broken arbitrarily. Afterwards, γ_j^{i+1} are updated for all these bins.

Below, we consider two subclasses of inputs for AAP, for which we show how to define the sets $S_1, \ldots, S_h$ and select the groups in line 7. We then prove that Algorithm 3 is guaranteed to succeed, yielding an $(1 + O(\varepsilon))$ approximation.

Instances with Unit-Size Items: Suppose that $s_i = 1$ for all $i \in [n]$. For $q \in [h]$ let S_q be the set of all groups for which the scaled and rounded group utility is $(1+\varepsilon)^q$. Since the value of the scaled solution is in $[1, n/\varepsilon]$, we have $h \leq \left\lceil \log_{(1+\varepsilon)} n/\varepsilon \right\rceil \leq \left\lceil (\ln n/\varepsilon) \cdot (\log_{(1+\varepsilon)} e) \right\rceil \leq (1/\varepsilon)\ln n/\varepsilon$.

For a choice of $w_1, \ldots, w_h$, for each S_q satisfying $w_q > 0$, $q \in [h]$, consider the groups in S_q in a nondecreasing order of their cardinality and add groups to the chosen subset S'_q one by one, until $P(S'_q) \geq w_q(\varepsilon n/h)$ for the first time. Order the resulting chosen set $\cup_{q=1}^{h} S'_q$ in a nondecreasing order of group cardinality.

Theorem 5.1 *Given an instance of* AAP *with unit size items, Algorithm 3 with the defined selection and ordering is guaranteed to succeed, yielding a feasible packing of utility at least* $(1 - O(\varepsilon))OPT$.

Proof: It suffices to show that there are guesses in lines 1 and 6 for which the algorithm produces a feasible packing of the desired utility. Consider the guess of $\mathcal{O}$ for which $\max\{P_{\max}, (1-\varepsilon)OPT\} \leq \mathcal{O} \leq OPT$. Let U be an optimal solution, and let $P(U \cap S_q)$ be the scaled and rounded overall utility of the groups in $U \cap S_q$. Set $w_q^* = \lfloor h \cdot P(U \cap S_q)/(\varepsilon n) \rfloor$. Consider the guess $w_1^*, \ldots, w_h^*$ in line 6. Suppose that $|U \cap S_q| = z$, and let $G_q^{*1}, \ldots, G_q^{*z}$ be the groups in $U \cap S_q$ ordered in nondecreasing order of their cardinality. Since all groups in S_q have the same (scaled and rounded) utility, and since the algorithm chooses the smallest number of groups whose total utility is at least $w_q^* \varepsilon n/h$, the algorithm chooses $z' \leq z$ groups. Let $G_i^1, \ldots, G_i^{z'}$ be the groups in S'_i ordered in nondecreasing order of their cardinality. Since the algorithm chooses the smallest cardinality groups, it follows that, for $j \in [z']$, $|G_q^j| \leq |G_q^{*j}|$. Thus, we can swap the groups $G_q^{*1} \ldots G_q^{*z'}$ by the groups $G_q^1, \ldots, G_q^{z'}$ and omit the groups $G_q^{*z'+1} \ldots G_q^{*z}$ from

the solution U without affecting its feasibility. We conclude that $\cup_{q\in[h]}S'_q$ is a feasible solution. Clearly, the (original unscaled) utility of this solution is $(1-O(\varepsilon))OPT$.

We need to show that the greedy procedure yields a feasible packing of $\cup_{q\in[h]}S'_q$. Let $\ell = |\cup_{q\in[h]} S'_q|$, and let $G_{\pi(1)},\ldots,G_{\pi(\ell)}$ be the groups in $\cup_{q\in[h]}S'_q$ ordered by their cardinality. We claim that the assignment by the greedy procedure is feasible.

Consider a feasible assignment of the groups in $\cup_{q\in[h]}S'_q$. (The existence of such an assignment is proved above.) Assume that t is the smallest index such that the feasible solution assigns items from $G_{\pi(t)}$ differently than the greedy procedure. Specifically, there are two bins j and j' such that $\gamma^t_j > \gamma^t_{j'}$, and the feasible solution assigns an item x from $G_{\pi(t)}$ to bin j', while it does not assign any item from $G_{\pi(t)}$ to bin j. We show that it is possible to reassign x from bin j' to bin j without violating the feasibility of the solution. If bin j is not full then item x can simply be moved to bin j. Otherwise, let B_j be the set of items assigned to bin j in the feasible solution that do not belong to groups $G_{\pi(1)},\ldots,G_{\pi(t-1)}$, and define $B_{j'}$ similarly. Since bin j is full, and the items are of unit size, $|B_j| = \gamma^t_j > |B_{j'}|$. Thus, B_j must contain an item y from some group G, such that no items from G are in $B_{j'}$. Clearly, swapping items x and y maintains the feasibility of the solution. We continue in the same manner until the feasible solution is the one produced by the greedy procedure. ∎

Instances with a Constant Number of Group Cardinalities and Item Sizes that Form a Divisible Sequence: Recall that a sequence $d_1 < d_2 < \cdots < d_D$ is *divisible* if d_{i-1} divides d_i for all $1 < i \le D$. Consider instances in which the different item sizes $\{s_i\}$ form a divisible sequence, and D, the number of different size values, may be arbitrary. This type of instance arises in our applications, as mentioned in Section 1.1. In addition, 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.

Let $h = r\cdot h'$ (for h' to be bounded below). For $j\in[r]$ and $i\in[h']$ define $S_{(j-1)\cdot h'+i}$ to be the set of all groups of cardinality $k(j)$ for which the scaled and rounded group utility is $(1+\varepsilon)^i$. Since the value of the scaled solution is in $[1,n/\varepsilon]$, we have $h \le r\left\lceil \log_{(1+\varepsilon)} n/\varepsilon \right\rceil \le (r/\varepsilon)\ln n/\varepsilon$. For convenience we use from now on a double index to refer to sets, where $S_{j,i}$ (and $w_{j,i}$) refers to $S_{(j-1)\cdot h'+i}$ ($w_{(j-1)\cdot h'+i}$, respectively).

For a guess of $w_{1,1},\ldots,w_{r,h'}$, for each $S_{j,i}$ for which $w_{j,i}>0$, $j\in[r]$ and $i\in[h']$, consider the groups in $S_{j,i}$ in a nondecreasing order of their item size, and add groups to the chosen subset $S'_{j,i}$ one by one until $P(S'_{j,i}) \ge w_{j,i}(\varepsilon n/h)$ for the first time. Order the resulting set of groups $\cup_{j\in[r],i\in[h']}S'_{j,i}$ in a nonincreasing order of their item size.

Theorem 5.2 *Given an instance of* AAP *with a constant number of group cardinalities and item sizes that form a divisible sequence, Algorithm 3 with the defined selection and ordering is guaranteed to succeed, yielding a feasible packing of utility at least* $(1-O(\varepsilon))OPT$.

Proof: It suffices to show that there are guesses in lines 1 and 6 for which the algorithm produces a feasible packing of the desired utility. Consider the guess of $\mathcal{O}$ for which $\max\{P_{\max},(1-\varepsilon)OPT\} \le \mathcal{O} \le OPT$. Let U be an optimal solution, and let $P(U\cap S_{j,i})$ be the scaled and rounded overall utility of the groups in $U\cap S_{j,i}$. Set $w^*_{j,i} = \lfloor h\cdot P(U\cap S_i)/(\varepsilon n)\rfloor$. Consider the guess $w^*_{1,1},\ldots,w^*_{r,h'}$ in line 6.

Suppose that $|U\cap S_{j,i}| = z$, and let Let $G^{*1}_{j,i},\ldots,G^{*z}_{j,i}$ be the groups in $U\cap S_{j,i}$ in nondecreasing order by their item size. Since all groups in $S_{j,i}$ have the same (scaled and rounded) utility, and since the algorithm chooses the smallest number of groups whose total utility is at least $w^*_{j,i}\varepsilon n/h$,

the algorithm chooses $z' \leq z$ groups. Let $G^1_{j,i}, \ldots, G^{z'}_{j,i}$ be the groups in $S'_{j,i}$ in nondecreasing order by their item size. Recall that the cardinality of all sets in $S_{j,i}$ is $k(j)$. Since the algorithm chooses the groups with the smallest item size, it follows that, for $\ell \in [z']$, $s^{\ell}_{j,i} \leq s^{*\ell}_{j,i}$. Thus, we can swap the groups $G^{*1}_{j,i} \ldots G^{*z'}_{j,i}$ by the groups $G^1_{j,i}, \ldots, G^{z'}_{j,i}$ and omit the groups $G^{*z'+1}_{j,i} \ldots G^{*z}_{j,i}$ from the solution U without affecting its feasibility. We conclude that $\cup_{j\in[r],i\in[h']} S'_{j,i}$ is a feasible solution. Clearly, the (original unscaled) utility of this solution is $(1 - O(\varepsilon))OPT$.

We need to show that the greedy procedure yields a feasible packing of $\cup_{j\in[r],i\in[h']} S'_{j,i}$. Let $\ell = |\cup_{j\in[r],i\in[h']} S'_{j,i}|$, and let $G_{\pi(1)}, \ldots, G_{\pi(\ell)}$ be the groups in $\cup_{j\in[r],i\in[h']} S'_{j,i}$ ordered in nonincreasing order of their item size. We claim that the assignment by the greedy procedure is feasible.

Consider a feasible assignment of the groups in $\cup_{j\in[r],i\in[h']} S'_{j,i}$. Assume that t is the smallest index such that the feasible solution assigns items from $G_{\pi(t)}$ differently than the greedy procedure. Specifically, there are two bins j and j', such that $\gamma^t_j > \gamma^t_{j'}$ and the feasible solution assigns an item x from $G_{\pi(t)}$ to bin j', while it does not assign any item from $G_{\pi(t)}$ to bin j. We show that it is possible to reassign x from bin j' without violating the feasibility of the solution. If bin j has enough free capacity to include x, we simply move x from bin j' to bin j. By our assumption, bin j was not assigned any item from group t earlier; therefore, this reassignment does not violate the condition that each bin is assigned at most one item from each group. Otherwise, let B_j be the set of items assigned to bin j in the feasible solution that do not belong to groups $G_{\pi(1)}, \ldots, G_{\pi(t-1)}$, and define $B_{j'}$ similarly. If the total size of the items in B_j is no larger than the total size of the items in $B_{j'}$ then swap the items in B_j and $B_{j'}$. Note that, since $\gamma^i_j \geq \gamma^i_{j'}$, both bins are still packed within capacity, and no bin is assigned two items from the same group. Suppose that the total size of the items in B_j is larger than the total size of the items in $B_{j'}$. Consider the subset $B'_j \subseteq B_j$ of items in B_j that belong to groups that are not assigned to bin j'. Since bin j' contains an item from $G_{\pi(t)}$, and bin j has no such item, the size of B'_j is at least $s_{\pi(t)}$. Also, since the groups are sorted in nonincreasing order of item size, the size of every item in B'_j is no more than $s_{\pi(t)}$. Since the item sizes form a divisible sequence, we can find a subset $B''_j \subseteq B'_j$ of items whose total size is exactly $s_{\pi(t)}$. This can be done simply by adding the items in B'_j to B''_j in nonincreasing order by size, until their total size is $s_{\pi(t)}$. Note that the fact that the item sizes form a divisible sequence implies that after we add an item of size s to B''_j, s divides the difference between $s_{\pi(t)}$ and the (current) total size of B''_j. This implies that at some point the total size of B''_j must be exactly $s_{\pi(t)}$. Now, swap the items in B''_j assigned to bin j with x in bin j'. This swap maintains feasibility. We continue in the same manner, until the feasible assignment is identical with the greedy one. ∎

References

[1] R. Adany, S. Kraus, and F. Ordonez. Allocation algorithms for personal tv advertisements. *Multimedia Systems*, 19:79–93, 2013.

[2] A. Ageev and M. Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal Combinatorial Optimization*, 8(3):307–328, 2004.

[3] 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.

[4] C. Chekuri and S. Khanna. On multi-dimensional packing problems. *SIAM J. on Computing*, 33(4):837–851, 2004.

[5] C. Chekuri and S. Khanna. A PTAS for the multiple knapsack problem. *SIAM J. on Computing*, 35(3):713–728, 2006.

[6] 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.

[7] U. Feige. A threshold of $ln\ n$ for approximating set cover. *J.of ACM*, 45(4):634–652, 1998.

[8] U. Feige and J. Vondrák. Approximation algorithms for allocation problems: Improving the factor of 1-1/e. In *FOCS*, pages 667–676, 2006.

[9] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko. Tight approximation algorithms for maximum separable assignment problems. *Math. Oper. Res.*, 36(3):416–431, 2011.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W.H. Freeman, New York, January 1979.

[11] Google AdWords. `http://adwords.google.com`.

[12] 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. *J. of Media Economics*, 19:55–79, 2006.

[13] A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *SODA*, pages 545–554, 2009.

[14] A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. *Mathematics of Operations Research*, 38(4):729–739, 2013.

[15] 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.

[16] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations.* 1990.

[17] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Math. Programming*, 14:265–294, 1978.

[18] Nielsen Media Research. Advertising fact sheet. blog.nielsen.com, September 2010.

[19] Nielsen Media Research. The cross-platform report, quarter 1, 2012 – US. blog.nielsen.com, May 2012.

[20] Nielsen Media Research. Nielsen's quarterly global adview pulse report. blog.nielsen.com, April 2012.

[21] Z. Nutov, I. Beniaminy, and R. Yuster. A (1-1/e)-approximation algorithm for the generalized assignment problem. *Operations Research Letters*, 34(3):283–288, 2006.

[22] 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.

[23] K. Pramataris, D. Papakyriakopoulos, G. Lekakos, and N. Mulonopoulos. Personalized Interactive TV Advertising: The iMEDIA Business Model. *Electronic Markets*, 11:1–9, 2001.

[24] D. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62(1):461–474, 1993.

[25] SintecMedia - On Air. `http://www.sintecmedia.com/OnAir.html`.

[26] M. Sviridenko. A note on maximizing a submodular set function subject to knapsack constraint. *Operations Research Letters*, 32:41–43, 2004.

[27] The Interactive Advertising Bureau (IAB). `http://iab.net`.

[28] C. Young. Why TV spot length matters. *Admap*, (497):45–48, September 2008.

A Hardness Results

In this section we analyze the hardness of several restricted instances of AGAP. First, note that AAP with unit-utilities, unit-group size, and no assignment restrictions yields an instance of MKP, that is known to be strongly NP-hard [16]. The presence of assignment restrictions makes the AAP problem with unit-size, unit-utility and uniform-cardinality, APX-hard, as shown in [21] for multiple knapsack with assignment restrictions.

The rest of our hardness results refer to restricted and natural instances of AGAP. We first show that AGAP is APX-hard already for very restricted instances.

Theorem A.1 *Unless $P = NP$,* AGAP *cannot be approximated within factor strictly better than $e/(e-1)$, 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 and then apply Feige's celebrated result [7] on the approximation threshold of maximum coverage. 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. Let $m = \max\{|U|/k, |S_1|, \ldots, |S_n|\}$. Define a bin of unit capacity for each element $j \in U$. If $mk > |U|$ define $mk - |U|$ additional "dummy" bins of unit capacity. Also, define a group of items, i, for each subset S_i, $1 \leq i \leq n$. The size of group i is m and it contains an item for each element of S_i and if $m > |S_i|$, $m - |S_i|$ additional "dummy" items. Consider an item ℓ in group i. If it is a "dummy" item then the utility of assigning it to any bin is 0. Otherwise, suppose that item ℓ corresponds to element $j \in S_i$. In this case the utility of assigning item ℓ to bin j is 1 and the utility of assigning it to any other bin (including the "dummy" bins) is 0.

Since the size of each group is m and we have exactly mk bins any k groups (or less) can be packed feasibly. Since the utility of each group is positive we can assume that the optimal solution packs k groups. Consider any k subsets $S_{\pi 1}, \ldots S_{\pi k}$ and the corresponding groups

$\pi 1, \ldots, \pi k$. To complete the proof we show in the next claim that the maximum utility that can be obtained by packing groups $\pi 1, \ldots, \pi k$ is the maximum coverage of the k subsets, i.e., $|\cup_{r=1}^{k} S_{\pi r}|$.

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

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

To achieve this utility we assign the items in group $\pi 1$ that correspond to elements of $S_{\pi 1}$ to the bins corresponding to these items and thus achieve utility 1 for each such item, We proceed in the same way with the items in group $\pi 2$ that correspond to elements of $S_{\pi 2} \setminus S_{\pi 1}$, and the with the items in group $\pi 3$ that correspond to elements of $S_{\pi 3} \setminus \{S_{\pi 1} \cup S_{\pi 2}\}$, and so on. Finally, we assign the rest of the items in these groups arbitrarily to empty bins. ■

By Claim A.1, 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. Thus, the hardness of approximation of maximum coverage implies the hardness of approximation of our problem. ■

Consider now a variant of AGAP in which we allow items of the same group to share a bin.

Theorem A.2 *If several items of the same group can be packed in the same bin, then* AGAP *cannot be approximated within any bounded factor, already in the case of a single group, two unit size bins and item utility that does not depend on a bin, unless $P = NP$.*

Proof: We show a straightforward reduction from PARTITION [10]. Given an instance $a_1, ..., a_n$ of PARTITION, one can generate an instance for AGAP with no assignment restrictions on items of the same group such that any bounded approximation to the AGAP variant implies a solution to the PARTITION instance. Consider an AGAP instance where there are only two unit size bins and one group G that contains n items of sizes $s_i = \frac{2a_i}{\sum_{j=1}^{n} a_j}$. Define the utility of each item to be $1/n$ independent of the bin to which it is assigned. Note that we can pack G iff there is partition of $\{a_i\}$ into two sets of the same size. In this case, we get a total utility of 1; otherwise, the utility is 0. Any polynomial time approximation algorithm for AGAP would enable us 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 (with no restriction on the maximum number of forbidden bins for an item). We show that the problem cannot be approximated within a polynomial in the minimum size of a group, even if the utility of the item is independent of the bin in which it is packed.

Theorem A.3 *Unless $NP = ZPP$,* AGAP *with forbidden bins cannot be approximated within a polynomial in the minimum size of a group, even if the utility of each item is uniform across all bins and the bins are unit size.*

Proof: By reduction from *packing integer program* (PIP) defined as follows. Given $A \in \{0,1\}^{m \times n}$, $b \in \mathbb{N}^m$ and $p \in [0,1]^n$ a PIP seeks to maximize $p^T x$ subject to $x \in \{0,1\}^n$ and $Ax \leq b$. Chekuri and Khanna [4] proved that PIP cannot be approximated within a factor of $\Omega(m^{1/(B+1)-\varepsilon})$, for any fixed integer B, unless $NP = ZPP$.

We show how to solve an instance of PIP via a reduction to AGAP with forbidden bins, where all the bins are of unit size. Define the groups $G_1, ..., G_n$, where G_j consists of the items $a_{i,j}$, for $i \in [m]$. Each item $a_{i,j}$ can be only in bin i (all the other bins are forbidden for $a_{i,j}$). The

size of $a_{i,j}$ is $A_{i,j}/b_i$ (which might be 0). Let the utility of an item of G_j be p_j/m. We note that each group can be packed in a unique way, therefore a feasible solution is determined only by the set $S \subseteq [n]$ of the groups that are packed feasibly. 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 and the total utility of the solution is $p^T x$.

Thus, the hardness of approximation of PIP implies the hardness of approximation of this variant of AGAP. Since the number of packing constraints translates to the size of the groups, m, we get that the it is hard to approximate this variant of AGAP within a factor of $\Omega(m^{1/(B+1)-\varepsilon})$, for any fixed integer B. ∎

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 A.4 *Unless $NP = ZPP$,* AGAP *with negative utilities cannot be approximated within a polynomial in the minimum size of a group, even if all bins are of unit size.*

We now turn to AGAP instances with arbitrary bin capacities.

Theorem A.5 *Unless $NP = ZPP$,* AGAP *with arbitrary bin capacities cannot be approximated within a polynomial in the minimum size of a group, even if the utility of each item is uniform across all bins.*

Proof: As in the proof of Theorem A.3 we use a reduction from PIP. Given an instance $A \in \{0,1\}^{m \times n}$, $b \in \mathbb{N}^m$ and $p \in [0,1]^n$, we construct the following AGAP instance. Each bin $i \in [m]$ has capacity $\frac{1}{(3n)^{i-1}\prod_{k=1}^{i-1}(b_k+\frac{1}{2})}$. The groups are $G_1, ..., G_n$, where G_j consists of the items $a_{i,j}$, for $i \in [m]$, each of utility p_j/m. Let the size of an item $a_{i,j}$ be $s_{i,j} = \frac{A_{i,j}+\frac{1}{2n}}{(3n)^{i-1}\prod_{k=1}^{i}(b_k+\frac{1}{2})}$.

We claim that $s_{i,j}$ is greater than the capacity of bin ℓ, for any $\ell > i$. To see this consider such a bin ℓ and compute the ratio of $s(i,j)$ to the capacity of bin ℓ. To lower bound this ratio assume $A_{i,j} = 0$ and $\ell = i+1$, we get that the ratio is at least $\frac{3n}{2n} > 1$. Hence, item $a_{i,j}$ can be packed only in bin i: it cannot be packed into bin ℓ for $\ell > i$, since it is too small, and it cannot be packed into bin ℓ, for $\ell < i$, as this bin must be occupied by item $a_{\ell,j}$. Let $S \subseteq [n]$ be a set of groups. S is a feasible solution iff for every bin i, $\sum_{j \in S} s_{i,j}$ is at most the capacity of bin i. This condition is equivalent to $\sum_{j \in S}(A_{i,j} + \frac{1}{2n}) \leq b_j + \frac{1}{2}$. Let x be the indicator vector of S. Because $A_{i,j} \in \{0,1\}$ and $|S| \leq n$, S is a feasible solution iff, for all i, $\sum_{j=1}^{n} A_{i,j}x_j \leq b_i$, i.e. x is a feasible solution of the PIP instance. Note that the objective function of both problems is $p^T x$. It follows that the hardness of approximation of PIP implies the hardness of approximation of this variant of AGAP. ∎