Generalized Assignment of Time-Sensitive Item Groups

Kanthi Sarpatwar
IBM Research, Yorktown Heights, NY, USA
sarpatwa@us.ibm.com

Baruch Schieber
IBM Research, Yorktown Heights, NY, USA
sbar@us.ibm.com

Hadas Shachnai
Computer Science Department, Technion, Haifa, Israel
hadas@cs.technion.ac.il

Abstract

We study the generalized assignment problem with time-sensitive item groups (χ-AGAP). It has central applications in advertisement placement on the Internet, and in virtual network embedding in Cloud data centers. We are given a set of items, partitioned into n groups, and a set of T identical bins (or, time-slots). Each group $1 \leq j \leq n$ has a time-window $\chi_j = [r_j, d_j] \subseteq [T]$ in which it can be packed. Each item i in group j has a size $s_i > 0$ and a non-negative utility u_{it} when packed into bin $t \in \chi_j$. A bin can accommodate at most one item from each group and the total size of the items in a bin cannot exceed its capacity. The goal is to find a feasible packing of a subset of the items in the bins such that the total utility from groups that are *completely* packed is maximized. Our main result is an $\Omega(1)$-approximation algorithm for χ-AGAP. Our approximation technique relies on a non-trivial rounding of a configuration LP, which can be adapted to other common scenarios of resource allocation in Cloud data centers.

2012 ACM Subject Classification Theory of computation → Packing and covering problems

Keywords and phrases Approximation Algorithms, Packing and Covering problems, Generalized Assignment problem

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2018.24

Acknowledgements H. Shachnai's work was conducted during a visit to DIMACS partially supported by the National Science Foundation under grant number CCF-1445755.

1 Introduction

In the classic *generalized assignment problem* (GAP), we are given a set of N items and T bins, $[T] = \{1, 2, \ldots, T\}$. Each item $i \in [N]$ has a size $s_i > 0$ and a non-negative utility u_{it} when packed into a bin $t \in [T]$.[1] The goal is to feasibly pack in the bins a subset of the items of maximum total utility. GAP has been widely studied, with applications ranging from manufacturing systems to regional planning (see, e.g., [3, 10]). In discrete optimization, GAP is well-known as a special case of the *separable assignment* problem and, more generally, of *submodular maximization* (see, e.g., [27, 15, 5, 6]). We consider an *all-or-nothing* variant of

[1] While item sizes may also depend on the bins, we focus here on the uniform case arising in our applications.

GAP, whose central applications include targeted advertising and virtual network embedding in Cloud data centers.

In *all-or-nothing* generalized assignment with time windows (χ-AGAP) we are given a set of T identical bins (or time-slots) and a set I of N items partitioned into n disjoint groups J. Each group $j \in J$ consists of a subset of items $I_j \subseteq I$ and has a time-window $\chi_j = [r_j, d_j] \subseteq [T]$ in which it can be packed. An item $i \in [N]$ which belongs to group j has a size $s_i > 0$ and a non-negative utility u_{it} when packed into bin $t \in \chi_j$. 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. The goal is to find a feasible packing of a subset of the items in the bins such that the total utility from groups that are *completely packed* is maximized.

As a primary motivation for our study, consider the management of an advertising campaign (i.e., a series of advertisements sharing a single theme) targeted at specific audience. Given such a large set of campaigns that can potentially be delivered to the media audience, a service provider attempts to fully deliver a subset of campaigns that maximizes the total revenue [26, 17, 21]. Each campaign is associated with a time-window (certain days/weeks, or hours in a given day) during which all of its ads need to be delivered. To increase the number of viewers exposed to an ad campaign, any continuous posting of ads (e.g., commercial break on TV) may contain a *single* ad from this campaign. Each ad has a given length (= size), which remains the same, regardless of the time slot in which it is posted, and possibly different revenues when placed in different breaks. The revenue from an ad campaign is the sum of the individual revenues of its ads, that is gained only if all of the campaign ads were assigned to breaks. Hence, a campaign scheduling instance can be cast as an instance of χ-AGAP.

Our problem has a central application also in *virtual network embedding (VNE)* in Cloud data centers, where a collection of virtual networks is mapped to a substrate *software defined network (SDN)* [28, 23, 20]. In the initial step of the VNE process, the nodes of each virtual network are mapped to nodes in the physical network. Each virtual network can be viewed as a *request* that is fulfilled only if all of its demands can be satisfied; else, the request is dropped. A common approach is to assign the nodes belonging to a single request to different substrate nodes and interconnect them by physical paths in the substrate network corresponding to the virtual links [9, 24]. Each node of a virtual network is associated with a resource demand (= size) and a revenue from assigning the node to a given physical node (= bin). The goal is to maximize the revenue gained from the VNE process. This yields an instance of χ-AGAP, where all groups (= virtual networks) share the same time window.

1.1 Related Work

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 $\rho \in (0,1]$, we say that $\mathcal{A}$ is a ρ-approximation algorithm if, for any instance I, $\frac{\mathcal{A}(I)}{OPT(I)} \geq \rho$.

Recall that the special case of χ-AGAP where all groups consist of a *single* item and have the same time window $[T]$ 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 [7]. Fleischer et al. [15] obtained a $(1-e^{-1})$-approximation for GAP, as a special case of the *separable assignment problem.* The best known ratio is $1 - e^{-1} + \varepsilon$ for some absolute constant $\varepsilon > 0$ [14].

Adany *et al.* [1] considered *all-or-nothing* GAP (AGAP), the special case of χ-AGAP where $\chi_j = [T]$ for all $j \in J$. They presented an $\Omega(1)$-approximation algorithm for general instances, and a $(1/3-\varepsilon)$-approximation for the special case where the utility of an item is identical across the bins, called the *group packing* (GP) problem. They also showed that unless $NP = ZPP$, GP cannot be approximated within a polynomial in the minimum size of a group, if we allow forbidden bins for the items. This implies that a polynomial time constant approximation for χ-AGAP with arbitrary forbidden bins is unlikely to exist. Indeed, our results rely strongly on the structural properties of the interval graph corresponding to the time windows which define the set of allowed bins for each group.

The Ad placement problem, introduced by Adler et al. [2], is the special case of χ-AGAP where all items in group $j \in J$ are identical, and each item has the same utility across the bins; also, all groups j share the same time window $\chi_j = [T]$. Freund and Naor [16] presented a $(1/3-\varepsilon)$-approximation for the problem. Ad placement has also been widely studied under different objectives (see, e.g., [12, 16, 13, 19, 18] and the comprehensive survey in [22]). In [25], we present approximation algorithms for more general scenarios of Ad placement, where groups may have different time windows, or ads are associated with multiple resource demands.

A related problem is *group packing of items into multiple knapsacks* (GMKP), introduced in [8]. In this generalization of *multiple knapsack* groups of items need to be placed in a set of bins. As in χ-AGAP, a revenue is gained for a group only if all of its items are packed. However, GMKP differs from χ-AGAP in several ways: (i) two items from the same group can be placed in the same bin. (ii) the utility of each item is the same across the bins, and (iii) all groups share the same time window, $[T]$. The paper [8] presents approximation algorithms for GMKP, assuming that the total size of the items in group j satisfies $\sum_{i \in I_j} s_i \leq \delta T$, for varying values of $\delta \in (0,1)$.

Our problem relates also to the *hypermatching assignment problem (HAP)* introduced in [11]. The input is a set of clients and a set of bundles of items. Each client has a budget; each bundle has a price and a profit depending on the client to which the bundle is assigned. Clients wish to buy one or more disjoint bundles of items . The goal is to maximize the total profit, while respecting client budgets. We can view the bundles as *groups*, and the clients as *time slots*, but then the remaining constraints are fairly different.

1.2 Contributions and Techniques

Our main results are constant factor approximation algorithms for χ-AGAP. Let $\chi_j = [r_j, d_j]$ denote the time window for processing group $j \in J$, and let $|\chi_j| = d_j - r_j + 1$ be the length of this interval. Throughout the paper, we assume that the time windows are large enough, namely, there is a constant $\lambda \in (0,1)$, such that $|I_j| \leq \lambda |\chi_j|$ for any group of items j. Such an assumption is quite reasonable in scenarios arising in our applications. We call λ the *slackness* parameter of the instance.

In Section 3, we present an $\Omega(1)$-approximation algorithm for *laminar* instances of χ-AGAP,[2] assuming that $\lambda < \frac{1}{5}$. We then extend the algorithm (in Section 4) to obtain an $\Omega(1)$-approximation for general χ-AGAP instances, assuming that $\lambda < \frac{1}{20}$. Thus, a main contribution of this paper is a general framework for obtaining a constant factor approximation based on the slackness of a given instance.

Techniques. Adany *et al.* [1] obtained a constant factor approximation algorithm for the special case where $\chi_j = [T] = \{1, 2, \ldots, T\}$, for all $j \in J$, assuming $\lambda < \frac{1}{2}$. We note that

[2] See the formal definition in Section 2.

even this special case is non-trivial and was solved via a novel reduction to maximizing a submodular function subject to a matroid constraint (over an exponential size universe of elements). An attempt to apply the same technique to the more general problem fails, due to the polynomially many knapsack constraints (representing the time windows for the n groups).

Thus, we apply a different approximation technique, which relies on a non-trivial rounding of the (fractional) solution for a configuration LP that guarantees a partial packing of a subset of groups.[3] This packing satisfies certain conditions, while still having a total utility that is within a constant factor from the optimal. Subsequently, using an elaborate evicting and repacking phase that exploits the structural properties of the instance, we obtain a feasible packing of *all* items in these groups.

We use our algorithm for laminar instances as a building block for solving general instances, via a transformation of a general instance to laminar. While the transformation is simple, applying the algorithm to the resulting laminar instance requires new ideas, as we must take into consideration the potential utility of each item when packed in its *original* time window (see Section 4). For the analysis we prove a general Repacking Theorem (see Section 3) stating the conditions required for obtaining a feasible solution from a *partial* packing of a given set of groups. This general theorem may be of independent interest in solving *all-or-nothing* variants of packing problems on interval graphs.

2 Preliminaries

We start with some definitions and notation. An instance of χ-AGAP consists of a set of identical (unit-sized) bins and a set of items I; each item $i \in I$ has a size $s_i > 0$ and utility u_{it} when packed into bin $t \in [T]$. The items are partitioned into n disjoint groups J. Each group $j \in J$ contains a subset of items $I_j \subseteq I$ which can be packed in the time window $\chi_j = [r_j, d_j] \subseteq T$.

A *feasible* packing of the subset $J' \subseteq J$ is an assignment of $|I_j|$ bins to every group $j \in J'$ such that the total size of items packed in any bin $t \in [T]$ is at most 1. Formally, $p : I' \to [T]$ is a feasible packing of the items in $I' = \bigcup_{j \in J'} I_j$ into bins if

1. For any $I_j \subseteq I'$, $|\bigcup_{i \in I_j} p(i)| = |I_j|$, i.e., the items in I_j are assigned to distinct bins.
2. For all $i \in I_j \subseteq I'$, $p(i) \in \chi_j$,
3. For all $t \in [T]$, $\sum_{\{i \in I' | p(i) = t\}} s_i \leq 1$.

The total utility of the packing p is given by $U_p = \sum_{j \in J'} \sum_{i \in I_j} u_{ip(i)}$. We say that $\lambda \in (0, 1)$ is the *slackness* parameter of a given instance if $|I_j| \leq \lambda |\chi_j|$ for all $j \in J$. Also, let $a_j = \sum_{i \in I_j} s_i$ denote the total size of items in group $j \in J$.

Finally, a set of intervals is *laminar* if for any two intervals χ_1 and χ_2, exactly one of the following holds: $\chi_1 \subseteq \chi_2$, $\chi_2 \subset \chi_1$ or $\chi_1 \cap \chi_2 = \emptyset$.

3 Approximation Algorithm for Laminar Instances

We first consider the case where $\mathcal{L} = \{\chi_j : j \in J\}$ forms a laminar family of intervals.

[3] Our rounding technique bears some similarities to the randomized rounding with alterations approach of [4].

Overview. Our algorithm proceeds in two phases. In the first phase, we find a subset of groups J' and items $I' \subseteq \bigcup_{j \in J'} I_j$ from these groups, along with a packing $p : I' \to [T]$ satisfying the following properties:

(1) For some constant $\alpha \in (0,1)$, for all $\chi \in \mathcal{L}$: $\sum_{j \in J' : \chi_j \subseteq \chi} a_j \leq \alpha|\chi|$. That is, the total size of *chosen* groups with time windows contained in χ is at most $\alpha|\chi|$.

(2) For any bin $t \in [T]$, the total size of items packed into t is at most 1, i.e., $\sum_{i \in I' : p(i) = t} s_i \leq 1$.

(3) Let OPT be the total profit of an optimal solution. Then, for some constant $\beta \in (0,1)$, $\sum_{i \in I'} u_{ip(i)} \geq \beta OPT$.

We note that, in this phase, some of the items in J' may not be packed into bins. The algorithm then moves to an *evict and repack* phase. Starting with a partial solution obtained from the first phase, we empty some of the bins. The eviction is performed in such a way that, for any interval $\chi \in \mathcal{L}$, at least a constant fraction of bins in χ are empty. This eviction step, which results in a constant factor loss in the approximation ratio, will facilitate the repacking of *all* items in the groups J'.

Phase 1. We start by solving a *linear programming (LP)* relaxation of our problem.

Configuration LP. For any group $j \in J$, a valid configuration M is an assignment of the items in group j into $|I_j|$ bins in χ_j. We can view a configuration for group j as a mapping $M : I_j \to \chi_j$, where $M(i)$ indicates the bin to which $i \in I_j$ is assigned. For a given group $j \in J$, let $\mathcal{C}_j$ be the set of all such valid configurations containing the items of group j. For a given bin $t \in [T]$, denote by $\mathcal{C}_t$ the set of valid configurations M in which there exists $i \in I$ such that $M(i) = t$. Define $\mathcal{C} = \bigcup_{j \in J} \mathcal{C}_j = \bigcup_{t \in [T]} \mathcal{C}_t$. Let x_M denote the indicator variable for choosing a configuration M. Throughout the discussion, each configuration M is (implicitly) associated with the assignment of the items of specific group $1 \leq j \leq n$. The utility of a configuration $M \in \mathcal{C}_j$ is given by $u_M = \sum_{i \in I_j} u_{iM(i)}$. Finally, for a given configuration, M, $i = M^{-1}(t)$ if i is assigned in M to bin t. The configuration LP can be stated as follows.

$$\text{Maximize} \left\{ \sum_{M \in \mathcal{C}} u_M x_M : \sum_{M \in \mathcal{C}_t} x_M s_{M^{-1}(t)} \leq 1 \; (\forall t \in [T]), \; \sum_{M \in \mathcal{C}_j} x_M \leq 1 \; (\forall j \in J), \; x_M \geq 0 \right\}$$

The first constraint ensures that the total size of items packed into a bin does not violate its capacity, and the second constraint implies that we choose at most one configuration for each group j. Note that the LP has exponentially many variables, and therefore we solve it using a dual separation oracle. The dual of the configuration LP is:

$$\text{Minimize} \left\{ \sum_{t \in [T]} y_t + \sum_{j \in J} z_j : z_j + \sum_{t \in \chi_j : M \in \mathcal{C}_t} y_t s_{M^{-1}(t)} \geq u_M \; (\forall j \in J, \; M \in \mathcal{C}_j), \; y_t, \, z_j \geq 0 \right\}$$

The dual program has exponentially many constraints. However, as shown below, it admits a separation oracle; thus, using the ellipsoid algorithm, it can be solved in polynomial time. Consider a dual solution $(y_t : t \in [T], \, z_j : j \in J)$. For each group $j \in J$, we construct a complete bipartite graph $H_j = (I_j, \chi_j, E_j)$, where $E_j = \{(i,t) : i \in I_j, t \in \chi_j\}$. For each item $i \in I_j$ and bin $t \in \chi_j$ we set $w(i,t) = u_{it} - y_t s_i$. Now, to test if there is a violating constraint corresponding to a configuration M for group j, we compute a maximum weight matching in H_j, denoted by M_j^*, in which every item $i \in I_j$ is matched.[4] Clearly, if there

[4] Since the edge weights in H_j may be negative, this can be done by a standard shift of edge weights, namely, setting $w'(i,j) = w(i,j) + |I_j|W$, where $W = \max_{(i,j) \in E_j} w(i,j)$, and solving maximum weight matching w.r.t. w'.

is a violating constraint corresponding to j, then the total weight for M_j^* has to satisfy $\sum_{(i,t)\in M_j^*} w(i,j) > z_j$.

Rounding Algorithm. Let $\pi_1, \pi_2 \in (0,1)$ be some constants (to be determined). We first solve the configuration LP to obtain an optimal fractional solution $(x_M^* : M \in \mathcal{C})$. Let $X_j = \sum_{M\in\mathcal{C}_j} x_M^*$. For each interval $\chi \in \mathcal{L}$, we associate a knapsack capacity of $\pi_2|\chi|$. We construct an ordering of groups satisfying the following property. If j and ℓ are two groups such that $\chi_j \subseteq \chi_\ell$, then j appears before ℓ in $\mathcal{O}$. In other words, we process groups in a *bottom up fashion* with respect to $\mathcal{L}$.

While considering a group j, we check if the knapsack capacity corresponding to χ_j has been violated. Formally, let J' denote the set of groups chosen before j; then, if $\sum_{j'\in J':\chi_{j'}\subseteq\chi_j} a_{j'} \le \pi_2|\chi_j|$, we add j to J' with probability $\pi_1 \sum_{M\in\mathcal{C}_j} x_M^* = \pi_1 X_j$; otherwise we discard j. Note that, to simplify the performance analysis, the knapsack constraint is checked *before* j is added, therefore the knapsack capacity may be slightly violated.

Let $I' \subseteq \cup_{j\in J'} I_j$ be the set of items in the groups of J' that have been packed so far, and suppose that j has been chosen, i.e., $j \in J'$. Now, we choose exactly one configuration for j with probability $\frac{x_M^*}{X_j}$. Denote the chosen configuration by M_j. For each $(i,t) \in M_j$, we now examine the total size of items packed in bin t. If $\sum_{i'\in I':p(i')=t} s_{i'} \le 1$ then we pack item i into bin t, i.e., $p(i) = t$. Note that this may result in violation of the capacity constraints in some bins. Thus, after processing all groups, we examine bins with violated capacities. Denote by S_t the set of currently packed items in such a bin t, and let i_t denote the last item packed into t. Before adding i_t, the capacity constraint was not violated; thus, $S_t \setminus \{i_t\}$ can be feasibly packed into t. We evict i_t from t if the combined profit of all the items in $S_t \setminus \{i_t\}$ is higher than that of i_t; otherwise, we evict all items except i_t from bin t. The pseudocode for the above rounding scheme is given in Algorithm 1.

Denote the *mean* (or *expected value*) of a random variable X by $\mathbb{E}[X]$, and the probability that an event E occurs by $\mathbb{P}(E)$. Note that Algorithm 1 is randomized; therefore, J', I' and p (and any function involving them) are random variables. The next lemma states some properties of the packing obtained in Phase 1.

▶ **Lemma 1.** *For suitable constants $0 < \pi_1 < \pi_2 < 1$ and $\lambda \in (0,1)$, assuming that for any group $j \in J$, $|I_j| \le \lambda|\chi_j|$, the following hold for Algorithm 1:*

(i) *The expected utility of the items in I' is at least a constant fraction of the optimal LP value, i.e.,*

$$\mathbb{E}\left[\sum_{i\in I'} u_{ip(i)}\right] \ge \frac{\pi_1}{2}\left(1 - \left(\frac{\pi_1}{\pi_2} + \pi_1\right)\right)\left[\sum_{M\in\mathcal{C}} u_M x_M^*\right]. \quad (1)$$

(ii) *For any interval $\chi \in \mathcal{L}$, the total size of groups chosen in χ is at most a constant fraction of $|\chi|$ i.e.,*

$$\sum_{j\in J':\chi_j\subseteq\chi} a_j \le (\pi_2 + \lambda)|\chi|. \quad (2)$$

(iii) *For any bin $t \in T$, the total size of items packed into t is within its capacity, i.e.,*

$$\sum_{i\in I':p(i)=t} s_i \le 1. \quad (3)$$

Note that properties (ii) and (iii) hold unconditionally (i.e., with probability 1).

Algorithm 1 Rounding the configuration LP solution.

```
Input: Laminar $\chi$-AGAP instance: $(I, J, T, \mathcal{L})$
Output: $J' \subseteq J$, $I' \subseteq \bigcup_{j \in J'} I_j$ and $p : I' \to [T]$
 1: Initialize $J' \leftarrow \emptyset$ and $I' \leftarrow \emptyset$. Solve the configuration LP and obtain an optimal fractional
    solution $(x^*_M : M \in \mathcal{C})$
 2: Let $\mathcal{O}$ be an ordering of groups $J$ as follows: for any $j, \ell \in J$, if $\chi_j \subseteq \chi_\ell$ then $j$ appears
    before $\ell$ in $\mathcal{O}$.
 3: for all $j$ in the ordered list $\mathcal{O}$ do
 4:     Let $X_j = \sum_{M \in \mathcal{C}_j} x^*_M$.
 5:     If $\sum_{j' \in J' : \chi_{j'} \subseteq \chi_j} a_{j'} \le \pi_2 |\chi_j|$ set $J' \leftarrow J' \cup \{j\}$ with probability $\pi_1 X_j$.
 6:     if $j \in J'$ then
 7:         Let $\mathcal{C}^+_j$ denote the configurations $M \in \mathcal{C}_j$ such that $x^*_M > 0$.
 8:         Choose exactly one configuration, $M_j \in \mathcal{C}^+_j$ with probability $\frac{x^*_M}{X_j}$ for configuration
    $M$.
 9:         For all $(i, t) \in M_j$, if $\sum_{i' \in I' : p(i') = t} s_{i'} \le 1$, set $I' \leftarrow I' \cup \{i\}$ and $p(i) = t$.
10:     end if
11: end for
12: for all $t \in [T]$ do
13:     if $\sum_{i' \in I' : p(i') = t} s_{i'} > 1$ then                ▷ handle violated bins
14:         Let $i$ be the last item with $p(i) = t$. If $u_{it} \ge \sum_{i' \neq i : p(i') = t} u_{i't}$: set $p(i') = \emptyset$ and
    $I' \leftarrow I' \setminus \{i'\}$ $\forall i' \neq i$; otherwise, set $p(i) = \emptyset$ and $I' \leftarrow I' \setminus \{i\}$.
15:     end if
16: end for
```

Proof. We start by lower bounding the expected utility obtained by using Algorithm 1 before eviction in Step 12.

(i) For any $t \in [T]$, let $U_t = \sum_{M \in \mathcal{C}} x^*_M \sum_{i \in I : (i,t) \in M} u_{it}$ denote the contribution of bin t to the optimal fractional solution value. It follows that $\sum_{t \in [T]} U_t = \sum_{M \in \mathcal{C}} x^*_M u_M$. Fix $t \in [T]$ and $i \in I$, and let j be the group containing item i. Let $\mathcal{C}_{i \to t}$ denote the set of all configurations M with $M(i) = t$. Let $\mathsf{iter}(j)$ be the iteration at which j is processed in Algorithm 1 (Step 3). We denote by $\mathbf{E}_{i \to t}$ the event "item i is assigned to bin t", i.e., $p(i) = t$. To show (1), it suffices to lower bound

$$\mathbb{E}\left[\sum_{i \in I' : p(i) = t} u_{it}\right] = \sum_{i \in I'} u_{it} \mathbb{P}(\mathbf{E}_{i \to t}),$$

for each $t \in [T]$. To this end, we define the following events:

($\mathbf{E}_1$) At the beginning of $\mathsf{iter}(j)$, $\sum_{j' \in J' : \chi_{j'} \subseteq \chi_j} a_{j'} \le \pi_2 |\chi_j|$.

($\mathbf{E}_2$) Group j is added to J'.

($\mathbf{E}_3$) At the beginning of $\mathsf{iter}(j)$, $\sum_{i' \in I' : p(i') = t} s_{i'} \le 1$.

We start by upper bounding the values $\mathbb{P}(\overline{\mathbf{E}_1})$ and $\mathbb{P}(\overline{\mathbf{E}_3})$. Since the probability of choosing a group j' is at most $\pi_1 X_{j'}$, using linearity of expectation, we have

$$\mathbb{E}\left[\sum_{j' \in J' : \chi_{j'} \subseteq \chi_j} a_{j'}\right] \le \sum_{j' \in J : \chi_{j'} \subseteq \chi_j} \mathbb{P}(j' \in J') a_{j'} \le \sum_{j' \in J : \chi_{j'} \subseteq \chi_j} \pi_1 X_{j'} a_{j'} \le \pi_1 |\chi_j|. \quad (4)$$

The last inequality follows from the constraints of the linear program, i.e., the total fractional size of groups packed into interval χ_j is at most $|\chi_j|$. Combining (4) with Markov's inequality, it follows that

$$\mathbb{P}(\overline{\mathbf{E}_1}) = \mathbb{P}\left(\sum_{j' \in J' : \chi_{j'} \subseteq \chi_j} a_{j'} > \pi_2 |\chi_j|\right) \le \frac{\pi_1}{\pi_2}. \tag{5}$$

Similarly, we have $\mathbb{E}\left[\sum_{i' \in I' : p(i') = t} s_{i'}\right] \le \sum_{M : (i', t) \in M} \pi_1 x^*_M \le \pi_1$; thus, by Markov's inequality,

$$\mathbb{P}(\overline{\mathbf{E}_3}) = \mathbb{P}\left(\sum_{i' \in I' : p(i') = t} s_{i'} > 1\right) \le \pi_1. \tag{6}$$

We have the following conditional probability inequalities:

$$\mathbb{P}(\mathbf{E}_2 | \mathbf{E}_1) = \pi_1 X_j \tag{7}$$

$$\mathbb{P}(\mathbf{E}_{i \to t} | \mathbf{E}_1 \cap \mathbf{E}_2 \cap \mathbf{E}_3) = \frac{\sum_{M \in \mathcal{C}_{i \to t}} x^*_M}{X_j} \tag{8}$$

Using (5) and (6),

$$\mathbb{P}(\mathbf{E}_1 \cap \mathbf{E}_3) = 1 - \mathbb{P}(\overline{\mathbf{E}_1} \cup \overline{\mathbf{E}_3}) \ge 1 - (\mathbb{P}(\overline{\mathbf{E}_1}) + \mathbb{P}(\overline{\mathbf{E}_3})) \ge 1 - (\frac{\pi_1}{\pi_2} + \pi_1) \tag{9}$$

Now, we compute the probability of $\mathbf{E}_{i \to t}$ as follows:

$$\begin{aligned}
\mathbb{P}(\mathbf{E}_{i \to t}) &= \mathbb{P}(\mathbf{E}_{i \to t} | \mathbf{E}_1 \cap \mathbf{E}_2 \cap \mathbf{E}_3)\mathbb{P}(\mathbf{E}_1 \cap \mathbf{E}_2 \cap \mathbf{E}_3) \\
&= \mathbb{P}(\mathbf{E}_{i \to t} | \mathbf{E}_1 \cap \mathbf{E}_2 \cap \mathbf{E}_3)\mathbb{P}(\mathbf{E}_2 | \mathbf{E}_1 \cap \mathbf{E}_3)\mathbb{P}(\mathbf{E}_1 \cap \mathbf{E}_3) \\
&= \mathbb{P}(\mathbf{E}_{i \to t} | \mathbf{E}_1 \cap \mathbf{E}_2 \cap \mathbf{E}_3)\mathbb{P}(\mathbf{E}_2 | \mathbf{E}_1)\mathbb{P}(\mathbf{E}_1 \cap \mathbf{E}_3)
\end{aligned}$$

The last equality follows from the fact that $\mathbb{P}(\mathbf{E}_2 | \mathbf{E}_1 \cap \mathbf{E}_3) = \mathbb{P}(\mathbf{E}_2 | \mathbf{E}_1)$. To see this, we note that by virtue of our algorithm $\mathbf{E}_2$ and $\mathbf{E}_3$ are conditionally independent events, given $\mathbf{E}_1$. Thus $\mathbb{P}(\mathbf{E}_2 | \mathbf{E}_1 \cap \mathbf{E}_3) = \mathbb{P}(\mathbf{E}_2 \cap \mathbf{E}_3 | \mathbf{E}_1)/\mathbb{P}(\mathbf{E}_3 | \mathbf{E}_1) = \mathbb{P}(\mathbf{E}_2 | \mathbf{E}_1)$. By the algorithm, $\mathbb{P}(\mathbf{E}_2 | \mathbf{E}_1) = \pi_1 X_j$. Thus, using (8) and (9), we have

$$\begin{aligned}
\mathbb{P}(\mathbf{E}_{i \to t}) &\ge \pi_1 X_j \left(\frac{\sum_{M \in \mathcal{C}_{i \to t}} x^*_M}{X_j}\right)\left(1 - \left(\frac{\pi_1}{\pi_2} + \pi_1\right)\right) \\
&= \pi_1 \left(\sum_{M \in \mathcal{C}_{i \to t}} x^*_M\right)\left(1 - \left(\frac{\pi_1}{\pi_2} + \pi_1\right)\right)
\end{aligned}$$

Thus, the expected utility for bin t is given by

$$\begin{aligned}
\mathbb{E}\left[\sum_{i \in I' : p(i) = t} u_{it}\right] &\ge \pi_1 \sum_{i \in I' : p(i) = t} u_{it} \left(\sum_{M \in \mathcal{C}_{i \to t}} x^*_{M}\right)\left(1 - \left(\frac{\pi_1}{\pi_2} + \pi_1\right)\right) \\
&= \pi_1 \left(1 - \left(\frac{\pi_1}{\pi_2} + \pi_1\right)\right)\left[\sum_{M \in \mathcal{C}} \sum_{i \in I : M(i) = t} u_{it} x^*_{M}\right]
\end{aligned}$$

Algorithm 2 Eviction Procedure.

```
Input: Packing $p : I' \to [T]$, a parameter $\pi_3 \in (0,1)$
Output: Set of evicted bins $E$ and a packing $p' : I'' \to [T] \setminus E$
1: Mark each interval $\chi \in \mathcal{L}$ as unprocessed. Initialize the set of evicted bins: $E \leftarrow \emptyset$, and
   let $I'' = I'$.
2: while $\exists$ a minimal unprocessed interval $\chi$ do
3:     if $|E \cap \chi| < \lceil (1-\pi_3)|\chi| \rceil$ then
4:         Sort the bins $\chi \setminus E$ in ascending order of the total profit of items packed in them.
5:         Let $E_\chi$ denote the first $\tau = \lceil (1-\pi_3)|\chi| \rceil - |E \cap \chi|$ bins in the above order.
6:         for all bins $t \in E_\chi$ do
7:             Evict the items in bin $t$, i.e., delete them from $I''$. Set $E \leftarrow E \cup \{t\}$.
8:         end for
9:         Mark $\chi$ as processed.
10:    end if
11: end while
12: Set $p'(i) = p(i)$, for all $i \in I''$
13: return $E$ and $p'$
```

Finally, by the linearity of expectation,

$$\begin{aligned}\mathbb{E}\left[\sum_{i \in I'} u_{ip(i)}\right] &= \sum_{t \in [T]} \mathbb{E}\left[\sum_{i \in I' : p(i) = t} u_{it}\right] \\ &\geq \sum_{t \in [T]} \pi_1 \left(1 - \left(\frac{\pi_1}{\pi_2} + \pi_1\right)\right) \left[\sum_{M \in \mathcal{C}} \sum_{i \in I : M(i) = t} u_{it} x^*_M\right] \\ &= \pi_1 \left(1 - \left(\frac{\pi_1}{\pi_2} + \pi_1\right)\right) \left[\sum_{M \in \mathcal{C}} u_M x^*_M\right]\end{aligned}$$

Now, for each bin t, let i_t be the last item, and S_t the set of all items packed into t. If the capacity of t has been violated, we note that each of the sets $S_t \setminus i_t$ and $\{i_t\}$ can be feasibly packed into t. We evict either i_t or $S_t \setminus i_t$, whichever is less profitable and thereby lose an additional factor of $\frac{1}{2}$ in the worst case.
The proofs of parts (ii) and (iii) are deferred to the Appendix. ◀

Phase 2. We now show that for suitable values of the parameters $0 < \pi_1 < \pi_2 < 1$, we can obtain a packing of *all* the items in J', such that the total profit is an $\Omega(1)$ fraction of the optimum. This yields a constant approximation for laminar χ-AGAP. Let $\pi_3 \in [2\lambda, 1)$ (to be determined).
We first *evict* the items from some "low" profit bins, such that for any interval $\chi \in \mathcal{L}$, there are at least $(1-\pi_3)|\chi|$ empty bins. (Recall that each interval χ can be viewed as a set of $|\chi|$ unit-sized bins.) Furthermore, the profit of items packed into bins that are not evicted is at least an $\Omega(1)$ fraction of the optimal solution value. Subsequently, we use the empty bins to repack all the unpacked items in the groups J'.
Eviction. Our eviction scheme is formally described in Algorithm 2. Bins are emptied while processing intervals in a *bottom-up fashion*, i.e., before an interval $\chi \in \mathcal{L}$ is processed, every sub-interval $\chi' \in \mathcal{L}$ where $\chi' \subset \chi$ is processed. Thus, in the main loop of the algorithm, we consider an unprocessed interval χ of minimum length in any given iteration (Step 2

Algorithm 3 Packing an item $i \in \hat{I}_j$.

```
1: If there exists a gray bin in avail(j) goto Step 2; otherwise, pick any white bin $t$ in
   avail(j) and pack $i$ into $t$. Color $t$ gray and exit. If no such white bin exists, report fail.
2: Let $t$ be a gray bin in avail(j), and $S_t$ the set of items packed into $t$. If $\sum_{i' \in S_t} s_{i'} + s_i \leq 1$,
   then pack $i$ into $t$ and exit; otherwise, goto Step 3.
3: Pick a white bin $t'$ in avail(j) and pack $i$ into $t'$. Color $t$ and $t'$ as black and pair up
   $t \leftrightarrow t'$. If no such white bin exists, report fail.
```

of Algorithm 2). Denote the current set of all evicted bins by E. We check if the total number of evicted bins contained in χ satisfies $|\chi \cap E| \geq \lceil (1 - \pi_3)|\chi| \rceil$. If not, we sort the bins $t \in \chi \setminus E$ in non-decreasing order of the total utility of items packed into them, given by $U_t = \sum_{\{i \in I' : p(i) = t\}} u_{it}$. We evict the first $\tau = \lceil (1 - \pi_3)|\chi| \rceil - |\chi \cap E|$ bins from $\chi \setminus E$.

The following lemma shows that at the end of this procedure we are left with at least a constant fraction of the original utility. We give the proof in the Appendix.

▶ **Lemma 2.** *Given a packing $p : I' \to [T]$ of items $I' \subseteq I$, let $p' : I'' \to [T] \setminus E$ be the packing obtained by applying Algorithm 2. Then,*

$$\sum_{i \in I''} u_{ip'(i)} \geq \frac{\pi_3}{2} \sum_{i \in I'} u_{ip(i)}.$$

Repacking. We now show that for any $\lambda \in (0, \frac{1}{5})$ and suitable values of $0 < \pi_1 < \pi_2 < 1$, and $\pi_3 \in [2\lambda, 1)$, we can always pack the items in J' into the empty bins obtained from the above eviction process. We prove a slightly more general result.

▶ **Theorem 3** (Repacking Theorem). *In a laminar instance of χ-AGAP, let $\hat{J}$ be a set of groups and $\hat{T} \subseteq [T]$ a subset of bins. Suppose that for parameters $\{\alpha, \beta, \gamma\} \in (0, 1)$ the following conditions are satisfied:*
(a) *For any $j \in \hat{J}$, $|\hat{I}_j| \leq \gamma |\chi_j|$*
(b) *For any $\chi \in \mathcal{L}$, there are at least $\alpha|\chi|$ bins in $\hat{T} \cap \chi$*
(c) *For any $\chi \in \mathcal{L}$, $\sum_{j \in \hat{J} : \chi_j \subseteq \chi} a_j \leq \beta |\chi|$.*
Then we can feasibly pack all the items $\hat{J}$ into $\hat{T}$ if $\gamma + 2\beta \leq \alpha$.

Proof. Let $S = \cup_{j \in \hat{J}} \hat{I}_j$ be the set of all items in the groups $\hat{J}$. In the course of our algorithm, we label bins with one of the possible three colors: *white, gray* and *black.* Our algorithm works in a *bottom-up* fashion and marks an interval χ *done* when it has successfully packed all the groups $j \in \hat{J}$ such that $\chi_j \subseteq \chi$. Initially all the bins are marked *white.* A bin t is marked *gray* when we pack an item into t, and black when we decide to add no more items to this bin. Consider an interval χ that has not been marked *done*, but every interval $\chi' \subset \chi$ is marked *done.* Let j be a group with $\chi_j = \chi$ that has not been packed completely. Pick an unpacked item $i \in I_j$. Let *avail*$(j) \subseteq \chi \cap \hat{T}$ be the set of bins that are not packed with items in group j. Algorithm 3 describes the formal procedure to pack item i.

We show that Algorithm 3 never reports **fail** and therefore feasibly packs all the items in S. Assume towards contradiction that the algorithm reports **fail** while packing an item i. Define a bin $t \in \chi \cap \hat{T}$ as *bad* if t is colored gray or some other item $i' \in \hat{I}_j$ has been packed in t. We first show that the following invariant holds, as long as no item in a group j^+ such that $\chi \subset \chi_{j^+}$ has been packed: the number of bad bins while processing group j is at most $\gamma|\chi|$. Assuming that the claim holds for each child interval of χ, namely, $\{\chi_1, \chi_2 \ldots, \chi_s\}$, before any group with time window χ is processed, we have the number of bad bins = number of gray bins is at most $\sum_{l \in [s]} \gamma|\chi_l| \leq \gamma|\chi|$. Now, consider the iteration where we pack some item $i \in \hat{I}_j$ into a bin t. If t is a gray bin, then the number of bad bins cannot increase. On the other hand, suppose t was white before packing i. If there are no gray bins in χ, then the number of bad bins is at most $|\hat{I}_j| \leq \gamma|\chi|$. Suppose there exist some gray bins, and consider those bins t' such that group j has no item packed into bin t'. If there are no such bins, then again the number of bad bins is at most $|\hat{I}_j| \leq \gamma|\chi|$. Otherwise, we must have considered one such gray bin t' and failed to pack i into it. By the virtue of the algorithm, we must have colored both t and t' black. Thus, the number of bad bins would not increase, and our claim holds. Now, since we pair the black bins $t \leftrightarrow t'$ only if $\sum_{i \in S_t} s_i + \sum_{i' \in S'_t} s_{i'} > 1$, the total number of black bins $< 2\beta|\chi|$. Hence, the total number of bins that are black or bad $< (\gamma + 2\beta)|\chi|$. Setting $\alpha \geq (\gamma + 2\beta)$, there should be at least one bin t^* that is neither black nor bad. But in this case, we could have packed i into t^* – a contradiction to the assumption the algorithm reports a **fail**. ◀

Putting it all together, we obtain the following main result.

▶ **Theorem 4.** *For any slackness parameter $\lambda \in (0, \frac{1}{5})$, there exists a polynomial time $\Omega(1)$-approximation algorithm for laminar χ-AGAP.*

Proof. Suppose we are given a laminar instance $\mathcal{I} = (I, J, T, \mathcal{L})$ of χ-AGAP with optimal profit OPT. We apply Lemma 1 to obtain a subset of groups J' and a *partial* packing p of a subset of items in J', denoted by I', such that

1. For any $\chi \in \mathcal{L}$, $\sum_{j \in J' : \chi_j \subseteq \chi} a_j \leq (\pi_2 + \lambda)|\chi|$
2. $\mathbb{E}\left[\sum_{i \in I'} u_{ip(i)}\right] \geq \frac{\pi_1}{2}\left(1 - \left(\frac{\pi_1}{\pi_2} + \pi_1\right)\right) OPT$.

Further, such a packing respects the capacity constraints and also ensures that no two items of the same group are packed in the same bin. Choosing $\pi_3 \geq 2\lambda$, we apply Algorithm 2 to compute a set of evicted bins E such that $|E \cap \chi| \geq \lceil (1 - \pi_3)|\chi| \rceil$, for any $\chi \in \mathcal{L}$. By Lemma 2 and property 2. above, the new packing $p' : I'' \rightarrow [T] \setminus E$ satisfies

$$\mathbb{E}\left[\sum_{i \in I''} u_{ip'(i)}\right] \geq \frac{\pi_3}{2}\mathbb{E}\left[\sum_{i \in I'} u_{ip(i)}\right] \geq \frac{\pi_1\pi_3}{4}\left(1 - \left(\frac{\pi_1}{\pi_2} + \pi_1\right)\right) OPT.$$

Let $\hat{I}_j = I_j \setminus I''$ be the unpacked items in group j, for all $j \in J'$, and let $\hat{J} = \cup_{j \in J'} \hat{I}_j$. Also, we can write $\pi_3 = 2\lambda + \pi'_3$, for $\pi'_3 \geq 0$. We show below that, for any $\lambda \in (0, \frac{1}{5})$, there exist positive constants π_1, π_2 and non-negative constant π'_3, such that (a) $\lambda + 2(\pi_2 + \lambda) \leq 1 - \pi_3$ and (b) $\frac{\pi_1}{\pi_2} + \pi_1 < 1$. With $\lambda \in (0, \frac{1}{5})$, we can select a positive constant value for π_2 and a non-negative value π'_3, such that $5\lambda \leq 1 - \pi'_3 - 2\pi_2$ (this is possible since $1 - 5\lambda > 0$). It follows that $5\lambda \leq 1 - (\pi_3 - 2\lambda) - 2\pi_2$, and (a) is satisfied. Now, setting $\pi_1 < \frac{1}{1 + 1/\pi_2}$, we ensure that (b) holds. To complete the proof, we show that the above implies that (i) all of the items in $\hat{J}$ can be packed into the empty bins in E, and (ii) the total expected utility is a constant fraction of the optimum. To show (i), we apply Theorem 3, setting $\hat{T} = E$, $\alpha = 1 - \pi_3$, $\beta = \pi_2 + \lambda$ and $\gamma = \lambda$. Then, from (a) we have that $\gamma + 2\beta \leq \alpha$. Thus, all of the conditions of Theorem 3 hold and we can pack all the items $\hat{J}$. Now, (ii) follows from the fact that $\frac{\pi_1\pi_3}{4}\left(1 - \left(\frac{\pi_1}{\pi_2} + \pi_1\right)\right)$ is a positive constant. Thus, we obtain a constant approximation for any $\lambda \in (0, \frac{1}{5})$. ◀

4 The General Case

In this section we extend our approach for laminar instances to the general case. As a first step, we transform the instance to a family of *laminar* intervals. Consider a general χ-AGAP

Algorithm 4 Transformation of a general family of intervals into a Laminar family.

```
Input: Job set $J$ and $\mathcal{W} = \{\chi_j : j \in J\}$
Output: Laminar family of intervals $\mathcal{L}$ and a mapping $\mathfrak{L} : \mathcal{W} \to \mathcal{L}$
1: Construct a binary tree with the interval $[T]$ as the root. Each node in the tree corresponds
   to an interval $\chi = [l, r] \subseteq [T]$.
2: If $r - l > 1$, then the node $[l, r]$ has two children corresponding to intervals $[l, \lfloor \frac{l+r}{2} \rfloor]$ and
   $[\lfloor \frac{l+r}{2} \rfloor + 1, r]$. We define $\mathcal{L}$ as the union of intervals corresponding to the nodes in the
   tree.
3: For each $\chi \in \mathcal{W}$, let $\chi'$ be the largest interval in $\mathcal{L}$ contained in $\chi$, breaking ties by
   picking the *rightmost* interval, then $\mathfrak{L}(\chi) = \chi'$.
```

instance. Let $\mathcal{W}$ denote the set of all time-windows for groups in J, i.e., $\mathcal{W} = \{\chi_j : j \in J\}$. We now construct a laminar family of intervals $\mathcal{L}$ and a mapping $\mathfrak{L} : \mathcal{W} \to \mathcal{L}$. Recall that $T = \max_{j \in J} d_j$.

► **Construction 5.** *The construction is formally described in Algorithm 4.*

It is natural to consider transforming a general instance to a laminar instance, using Algorithm 4, and then applying the LP rounding procedure (of Section 3) to the laminar instance. However, this can lead to a solution that is far from the optimal. Indeed, as item utilities depend on the bins, the configurations for the general instance can differ significantly from those of the laminar instance. Specifically, it may be the case that for some group $j \in J$, item $i \in I_j$ has high utility when packed in a bin $t \in \chi_j$, but $t \notin \mathfrak{L}(\chi_j)$. We overcome this issue by defining the configurations on the *original* intervals $\mathcal{W}$, while setting the knapsack constraints (in Step. 5 of Algorithm 1) on the intervals $\mathcal{L}$.

We give the details of the algorithm in the Appendix.

References

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

2 Micah Adler, Phillip B Gibbons, and Yossi Matias. Scheduling space-sharing for internet advertising. *Journal of Scheduling*, 5(2):103–119, 2002.

3 V. Balachandran. An integer generalized transportation model for optimal job assignment in computer networks. *Operations Research*, 24(4):742–759, 1976.

4 Nikhil Bansal, Nitish Korula, Viswanath Nagarajan, and Aravind Srinivasan. Solving packing integer programs via randomized rounding with alterations. *Theory of Computing*, 8(1):533–565, 2012.

5 Marco Bender, Clemens Thielen, and Stephan Westphal. Packing items into several bins facilitates approximating the separable assignment problem. *Information Processing Letters*, 115(6-8):570–575, 2015.

6 Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. In *SODA*, pages 392–403, 2016.

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

8 Lin Chen and Guochuan Zhang. Packing groups of items into multiple knapsacks. In *STACS*, pages 28:1–28:13, 2016.

9 NM Mosharaf Kabir Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Virtual network embedding with coordinated node and link mapping. In *INFOCOM*, pages 783–791. IEEE, 2009.

10 Robert G Cromley and Dean M Hanink. Coupling land use allocation models with raster GIS. *Journal of Geographical Systems*, 1(2):137–153, 1999.

11 Marek Cygan, Fabrizio Grandoni, and Monaldo Mastrolilli. How to sell hyperedges: The hypermatching assignment problem. In *SODA*, pages 342–351. SIAM, 2013.

12 Milind Dawande, Subodha Kumar, and Chelliah Sriskandarajah. Performance bounds of algorithms for scheduling advertisements on a web page. *Journal of Scheduling*, 6(4):373–394, 2003.

13 Milind Dawande, Subodha Kumar, and Chelliah Sriskandarajah. Scheduling web advertisements: a note on the minspace problem. *Journal of Scheduling*, 8(1):97–106, 2005.

14 Uriel Feige and Jan Vondrák. Approximation algorithms for allocation problems: Improving the factor of 1-1/e. In *FOCS*, pages 667–676, 2006.

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

16 Ari Freund and Joseph Naor. Approximating the advertisement placement problem. *Journal of Scheduling*, 7(5):365–374, 2004.

17 Francesca Guerriero, Giovanna Miglionico, and Filomena Olivito. Managing tv commercials inventory in the italian advertising market. *International Journal of Production Research*, 54(18):5499–5521, 2016.

18 Arshia Kaul, Sugandha Aggarwal, Anshu Gupta, Niraj Dayama, Mohan Krishnamoorthy, and PC Jha. Optimal advertising on a two-dimensional web banner. *International Journal of System Assurance Engineering and Management*, pages 1–6, 2017.

19 Subodha Kumar, Milind Dawande, and Vijay Mookerjee. Optimal scheduling and placement of internet banner advertisements. *IEEE Transactions on Knowledge and Data Engineering*, 19(11), 2007.

20 Amin Ghalami Osgouei, Amir Khorsandi Koohanestani, Hossein Saidi, and Ali Fanian. Online assignment of non-SDN virtual network nodes to a physical SDN. *Computer Networks*, 129:105–116, 2017.

21 Mark J Panaggio, Pak-Wing Fok, Ghan S Bhatt, Simon Burhoe, Michael Capps, Christina J Edholm, Fadoua El Moustaid, Tegan Emerson, Star-Lena Estock, Nathan Gold, Ryan Halabi, Madelyn Houser, Peter R Kramer, Hsuan-Wei Lee, Qingxia Li, Weiqiang Li, Dan Lu, Yuzhou Qian, Louis F Rossi, Deborah Shutt, Vicky Chuqiao Yang, and Yingxiang Zhou. Prediction and optimal scheduling of advertisements in linear television. *arXiv preprint arXiv:1608.07305*, 2016.

22 Shinjini Pandey, Goutam Dutta, and Harit Joshi. Survey on revenue management in media and broadcasting. *Interfaces*, 47(3):195–213, 2017.

23 Adil Razzaq, Peter Sjödin, and Markus Hidell. Minimizing bottleneck nodes of a substrate in virtual network embedding. In *NOF*, pages 35–40, 2011.

24 Robert Ricci, Chris Alfeld, and Jay Lepreau. A solver for the network testbed mapping problem. *ACM SIGCOMM Computer Communication Review*, 33(2):65–81, 2003.

25 Kanthi K. Sarpatwar, Baruch Schieber, and Hadas Shachnai. Brief announcement: Approximation algorithms for preemptive resource allocation. *To appear in SPAA*, 2018.

26 SintecMedia - On Air. http://www.sintecmedia.com/OnAir.html, 2013.

27 Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, pages 67–74, 2008.

28 Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *Computer Communication Review*, 38(2):17–29, 2008.

A Some Proofs

Proof of Lemma 1 (parts (ii) and (iii)):

(ii) We show (2) using induction on the intervals χ.
Base Step. Let χ be an interval containing no other interval in $\mathcal{L}$, i.e., there is no $\chi' \in \mathcal{L}$ such that $\chi' \subset \chi$. Let j be the last group with time window $\chi_j = \chi$, in the ordering $\mathcal{O}$, that is successfully added to J'; if there is no such group, we are done. Consider the iteration (in Algorithm 1, Step 3.) in which j was considered. In this iteration, we must have that $\sum_{j' \in J' : \chi_{j'} \subseteq \chi} a_{j'} \leq \pi_2 |\chi|$ – otherwise, j should have been discarded. Thus, upon adding j to J', we have

$$\sum_{j' \in J' : \chi_{j'} \subseteq \chi} a_{j'} \leq \pi_2 |\chi| + a_j \leq \pi_2 |\chi| + \lambda |\chi_j| = (\pi_2 + \lambda)|\chi|.$$

The second inequality holds since $a_j = \sum_{i \in I_j} s_i \leq |I_j| \leq \lambda |X_j|$.
Inductive Step. Let χ be an interval with child intervals $\chi_1, \chi_2, \ldots, \chi_s$ such that the claim holds for each χ_k, $k \in [s]$. First, suppose there is no group $j \in J'$ with $\chi_j = \chi$. In this case, we have

$$\sum_{j' \in J' : \chi_{j'} \subseteq \chi} a_{j'} = \sum_{k \in [s]} \sum_{j' \in J' : \chi_{j'} \subseteq \chi_k} a_{j'} \leq \sum_{k \in [s]} (\pi_2 + \lambda)|\chi_k| \leq (\pi_2 + \lambda)|\chi|.$$

Now, consider the case where there is a group $j \in J'$ with $\chi_j = \chi$ and w.l.o.g. let j be the last such group. As in the base case, in the iteration where j was added to J', we have $\sum_{j' \in J' : \chi_{j'} \subseteq \chi} a_{j'} \leq \pi_2 |\chi|$. Thus, after adding j to J', we have $\sum_{j' \in J' : \chi_{j'} \subseteq \chi} a_{j'} \leq (\pi_2 + \lambda)|\chi|$.
(iii) As shown in part (i) of the proof, for any bin t, we either keep the last item i_t, or the rest of the items, $S_t \setminus i_t$, and both respect the capacity constraint corresponding to t. ◀

Proof of Lemma 2:

Let $U_t = \sum_{i \in I' : p(i) = t} u_{it}$ denote the total utility of items in I' packed into t under the packing p. Using induction, we show that: in the iteration where an interval χ is marked processed, the total utility of items packed, under p', into χ is at least $\sum_{t \in \chi} \frac{\pi_3}{2} U_t$. We note that this claim may not hold for χ in subsequent iterations, when we process a parent interval $\chi^+ \supset \chi$. However, at the end of all iterations, this claim would ensure that under p' we are left with at least a constant fraction of the total profit under p, i.e., $\sum_{t \in [T]} \sum_{i \in I'' : p'(i) = t} u_{it} \geq \sum_{t \in [T]} \frac{\pi_3}{2} U_t$.
Base Case. Let $\chi \in \mathcal{L}$ be a leaf interval i.e., there is no $\chi' \in \mathcal{L}$ such that $\chi' \subset \chi$. Note that we process χ before any of its parent intervals $\chi^+ \supset \chi$. When χ is marked as *processed*, we would have evicted at most $\lceil (1 - \pi_3)|\chi| \rceil$ least profitable bins. Further,

$$\begin{aligned}\lceil (1 - \pi_3)|\chi| \rceil &\leq (1 - \pi_3)|\chi| + 1 \leq (1 - \pi_3)|\chi| + \frac{\pi_3}{2\lambda} \\ &\leq (1 - \pi_3)|\chi| + \frac{\pi_3}{2}|\chi| = (1 - \frac{\pi_3}{2})|\chi|. \qquad (10)\end{aligned}$$

The second inequality follows from the choice of $\pi_3 \geq 2\lambda$. For the last inequality, we note that there exists a group $j \in J$ with $\chi_j = \chi$ and therefore $1 \leq I_j \leq \lambda |\chi|$. Thus, in this case, at least $\frac{\pi_3}{2}|\chi|$ of the most profitable bins are not evicted, and the claim holds.
Inductive Step. Suppose χ is a non-leaf interval with children $\chi_l, l \in [s]$. We consider the iteration in which χ was marked *processed.* First, we observe that, at this iteration, the total number of bins that are not evicted in χ is at least $\frac{\pi_3}{2}|\chi|$. To see this, consider the following two cases:

(i) If some bins are evicted in the iteration where χ is being processed (Step 3. of Algorithm 2), then the number of evicted bins in χ is exactly $\lceil (1-\pi_3)|\chi| \rceil \le (1-\frac{\pi_3}{2})|\chi|$ (by Equation (10).

(ii) If no new bin is evicted in the iteration then, by the algorithm and Equation (10), the total number of evicted bins in χ is bounded by $\sum_{l\in[s]}\lceil (1-\pi_3)|\chi_l| \rceil \le \sum_{l\in[s]}(1-\frac{\pi_3}{2})|\chi_l| \le (1-\frac{\pi_3}{2})|\chi|$.

We use a charging argument to prove our original claim, i.e.,

$$\sum_{t\in\chi}\sum_{\{i\in I'':p'(i)=t\}} u_{it} \ge \frac{\pi_3}{2}\sum_{t\in\chi} U_t.$$

Every bin (evicted or otherwise) $t\in\chi$ is charged to a bin $t'\in\chi\setminus E$ that is not evicted, such that

1. the total profit of items packed in t' is greater than those packed in t, i.e., $U_{t'} \ge U_t$.
2. no bin t' is charged with more than $\frac{2}{\pi_3}$ bins.

This would clearly imply that the total utility of non-evicted bins is at least $\frac{\pi_3}{2}$ fraction of $\sum_{t\in\chi} U_t$. The charging is again done in a bottom-up fashion. For a leaf interval: a non-evicted bin is charged to itself; evicted bins are arbitrarily distributed among the non-evicted bins in such a way that no bin is charged with more than $\frac{2}{\pi_3}$ bins. This is possible because there are at least $\frac{\pi_3}{2}|\chi|$ bins that are not evicted.

Suppose that all the bins in each child interval, $\chi_l : l\in[s]$, of χ has been charged. When considering χ, some new bins may be evicted. Consider any newly evicted bin t that in turn might have been charged with a certain subset of evicted bins. We observe that all the bins in χ that are not evicted must have profit at least as much as t and are therefore more profitable than the evicted bins charged to t. Thus, we can arbitrarily assign these evicted bins (bins charged to t, including itself) to the non-evicted bins, as long as the number of bins charged to any bin is at most $\frac{2}{\pi_3}$. This is possible because we have at most $(1-\frac{\pi_3}{2})|\chi|$ evicted bins in χ at this iteration (i.e., when χ was marked processed). The claim follows by induction.

B Approximating General χ-AGAP instances

We describe below the extension of our algorithm for laminar instances to handle general χ-AGAP instances.

Phase 1. The LP remains the same. Note that the configurations are defined on the set of intervals $\mathcal{W}$

$$\text{Maximize}\left\{\sum_{M\in\mathcal{C}} u_M x_M : \sum_{M\in\mathcal{C}_t} x_M s_{M^{-1}(t)} \le 1\,(\forall t\in[T]),\ \sum_{M\in\mathcal{C}_j} x_M \le 1\,(\forall j\in J),\ x_M\ge 0\right\}$$

Rounding Algorithm. As in the laminar case, fix suitable constants $\pi_1,\pi_2\in(0,1)$ and solve the configuration LP to obtain an optimal fractional solution $(x^*_M : M\in\mathcal{C})$. Now, using Algorithm 4, transform $\mathcal{W}$ into a family of laminar intervals $\mathcal{L}$ and obtain a mapping $\mathfrak{L}:\mathcal{W}\to\mathcal{L}$ of intervals in $\mathcal{W}$ to those in $\mathcal{L}$. Next, we associate each interval $\chi\in\mathcal{L}$ with a knapsack capacity of $\pi_2|\chi|$. Further, the ordering of groups $\mathcal{O}$ is based on $\mathfrak{L}$ (and $\mathcal{L}$). That is, if j and ℓ are two groups such that $\mathfrak{L}(\chi_j)\subseteq\mathfrak{L}(\chi_\ell)$, then j appears before ℓ in $\mathcal{O}$.

While considering group j, we check if the knapsack capacity corresponding to $\mathfrak{L}(\chi_j)$ has been violated. Formally, let J' be the set of groups chosen before j; then, we check if

Algorithm 5 Rounding the configuration LP.

```
Input: A $\chi$-AGAP instance: $(I, J, T, \mathcal{W})$
Output: $J' \subseteq J$, $I' \subseteq \bigcup_{j \in J'} I_j$ and $p : I' \to [T]$
 1: Initialize $J' \leftarrow \emptyset$ and $I' \leftarrow \emptyset$.
 2: Solve the configuration LP and obtain an optimal fractional solution $(x^*_M : M \in \mathcal{C})$
 3: Compute an ordering $\mathcal{O}$ of groups in $J$ as follows: for any $j, \ell \in J$, if $\mathfrak{L}(\chi_j) \subseteq \mathfrak{L}(\chi_\ell)$ then
    $j$ appears before $j'$ in $\mathcal{O}$
 4: for all $j$ in accordance with the ordering $\mathcal{O}$ do
 5:     Let $X_j = \sum_{M \in \mathcal{C}_j} x^*_M$.
 6:     $\sum_{j' \in J' : \mathfrak{L}(\chi_{j'}) \subseteq \mathfrak{L}(\chi_j)} a_{j'} \leq \pi_2 |\mathfrak{L}(\chi_j)|$, set $J' \leftarrow J' \cup \{j\}$ with probability $\pi_1 X_j$.
 7:     If $j \in J'$, let $\mathcal{C}^+_j$ denote the configurations $M \in \mathcal{C}_j$ such that $x^*_M > 0$.
 8:     Choose exactly one configuration, $M_j \in \mathcal{C}^+_j$ with probability $\frac{x^*_M}{X_j}$ for configuration $M$.
 9:     For all $(i, t) \in M_j$, if $\sum_{i' \in I' : p(i') = t} s_{i'} \leq 1$, set $I' \leftarrow I' \cup \{i\}$ and $p(i) = t$.
10: end for
11: for all $t \in [T]$ do
12:     if $\sum_{i' \in I' : p(i') = t} s_{i'} > 1$ then                    ▷ handle violated bins
13:         Let $i$ be the last item with $p(i) = t$. If $u_{it} \geq \sum_{i' : p(i') = t} u_{i't}$: set $p(i') = \emptyset$ and
    $I' \leftarrow I' \setminus \{i'\}$ $\forall i' \neq i$; Otherwise, set $p(i) = \emptyset$ and $I' \leftarrow I' \setminus \{i\}$.
14:     end if
15: end for
```

$\sum_{j' \in J' : \mathfrak{L}(\chi_{j'}) \subseteq \mathfrak{L}(\chi_j)} a_{j'} \leq \pi_2 |\mathfrak{L}(\chi_j)|$. If this constraint holds, we add j into J' with probability $\pi_1 \sum_{M \in \mathcal{C}_j} x^*_M = \pi_1 X_j$, where $X_j = \sum_{M \in \mathcal{C}_j} x^*_M$; otherwise, we discard j. If j has been selected, i.e., $j \in J'$, then choosing a configuration and then picking a subset of items in I_j that can be packed is exactly the same as in the laminar case. Our rounding scheme is formally described in Algorithm 5.

The proof of the following lemma is similar to the proof of Lemma 1 (details omitted).

▶ **Lemma 6.** *For suitable constants π_1, π_2, λ, assuming that for any group $j \in J$, $|I_j| \leq \lambda |\chi_j|$, the following hold for Algorithm 5:*

(i) *The expected utility of the items in I' is at least a constant fraction of the optimal LP value, i.e.,*

$$\mathbb{E}\left[\sum_{i \in I'} u_{ip(i)}\right] \geq \frac{\pi_1}{2}\left(1 - \left(\frac{4\pi_1}{\pi_2} + \pi_1\right)\right)\left[\sum_{M \in \mathcal{C}} u_M x^*_M\right]$$

(ii) *For any interval $\chi \in \mathcal{L}$, the total size of groups chosen in χ is at most a constant fraction of $|\chi|$, i.e.,*

$$\sum_{j \in J' : \mathfrak{L}(\chi_j) \subseteq \chi} a_j \leq (\pi_2 + 4\lambda)|\chi|$$

(iii) *For any bin $t \in T$, the total size of items packed into t is within its capacity, i.e.,*

$$\sum_{i \in I' : p(i) = t} s_i \leq 1$$

Phase 2. We now apply the eviction and repacking phase on the transformed instance, i.e., considering the time windows $\chi \in \mathcal{L}$. We summarize in the next theorem.

We use in the analysis the next result (due to [25]). We include a proof for completeness.

▶ **Lemma 7.** *In Algorithm 4, the following properties hold:*

1. *For any $j \in J$, $|\chi_j| \leq 4|\mathfrak{L}(\chi_j)|$*
2. *For $\chi \in \mathcal{L}$, let $\tilde{\chi} = \{t \in \chi_j : j \in J,\ \mathfrak{L}(\chi_j) = \chi\}$, i.e., the union of all time-windows in $\mathcal{W}$ that are mapped to χ. Then, $|\tilde{\chi}| \leq 4|\chi|$.*

Proof. To prove the first property, it suffices to show that χ_j cannot completely contain 3 consecutive intervals in $\mathcal{L}$ that are at the same level as $\mathfrak{L}(\chi_j)$. Indeed, this would imply that χ_j cannot intersect more than 4 consecutive intervals, and therefore $|\chi_j| \leq 4|\mathfrak{L}(\chi_j)|$. Now, suppose χ_j contains at least 3 such consecutive intervals. Then, by the virtue of our algorithm, $\mathfrak{L}(\chi_j)$ is the rightmost interval. Let $\hat{\chi}$ be the parent of $\mathfrak{L}(\chi_j)$. Two cases arise:

Case 1: $\mathfrak{L}(\chi_j)$ is a left child of $\hat{\chi}$. Consider the two other consecutive intervals at the same level as $\mathfrak{L}(\chi_j)$ that are contained in χ_j. Observe that these two intervals are siblings; therefore, their parent (which is also in $\mathcal{L}$) is also contained in χ_j. This is a contradiction to the assumption that $\mathfrak{L}(\chi_j)$ is the largest interval in $\mathcal{L}$ contained in χ_j.

Case 2: $\mathfrak{L}(\chi_j)$ is a right child of $\hat{\chi}$. We observe that the sibling of $\mathfrak{L}(\chi_j)$ must also be contained in χ_j, implying that $\hat{\chi}$ is contained in χ_j, a contradiction.

We now prove the second property. For any $\chi = [s, d] \in \mathcal{L}$, let $\chi_l = [s_l, d_l] \in \mathcal{W}$ (resp. $\chi_r = [s_r, d_r]$) be the leftmost (resp. rightmost) interval in $\mathcal{W}$ such that $\mathfrak{L}(\chi_l) = \chi$ (resp. $\mathfrak{L}(\chi_r) = \chi$); then, $\tilde{\chi} = [s_l, d_r]$. Consider the intervals $\chi_1 = [s_l, s]$ and $\chi_2 = [d, d_r]$. As argued above, χ_l cannot contain 3 consecutive intervals in $\mathcal{L}$ at the same level as χ Thus, $|\chi_1| < 2|\chi|$.

Also, $|\chi_2| < |\chi|$; otherwise, there is an interval to the right of χ of the same size that can be mapped to χ_r. Thus, $|\chi_1| + |\chi_2| < 3|\chi|$. Now, the claim follows by observing that $|\tilde{\chi}| = |\chi_1| + |\chi| + |\chi_2| \leq 4|\chi|$. ◀

▶ **Theorem 8.** *For any slackness parameter $\lambda \in (0, \frac{1}{20})$, there exists a polynomial time $\Omega(1)$-approximation algorithm for general χ-AGAP instances.*

Proof. Suppose that the given instance $\mathcal{I} = (I, J, T, \mathcal{W})$ has an optimal profit OPT. We apply Lemma 6 to obtain a subset of groups J' and a *partial* packing p of a subset of items in J', denoted by I', such that:

1. For any $\chi \in \mathcal{L}$, $\sum_{j \in J' : \mathfrak{L}(\chi_j) \subseteq \mathfrak{L}(\chi)} a_j \leq (\pi_2 + 4\lambda)|\chi|$
2. $\mathbb{E}\left[\sum_{i \in I'} u_{ip(i)}\right] \geq \frac{\pi_1}{2}\left(1 - \left(\frac{4\pi_1}{\pi_2} + \pi_1\right)\right) OPT$.

Further, such a packing respects the capacity constraints and also ensures that no two items of the same group are packed in the same bin. Choosing a suitable $\pi_3 \geq 8\lambda$, we apply Algorithm 2 on the intervals $\mathcal{L}$ to compute a set of evicted bins E such that $|E \cap \chi| \geq (1 - \pi_3)|\chi|$, for any $\chi \in \mathcal{L}$. By Lemma 2 and property 2. above, the new packing $p' : I'' \to [T] \setminus E$ satisfies

$$\mathbb{E}\left[\sum_{i \in I''} u_{ip'(i)}\right] \geq \frac{\pi_3}{2}\mathbb{E}\left[\sum_{i \in I'} u_{ip(i)}\right] \geq \frac{\pi_1\pi_3}{4}\left(1 - \left(\frac{4\pi_1}{\pi_2} + \pi_1\right)\right) OPT.$$

Let $\hat{I}_j = I_j \setminus I''$ be the unassigned items in group j, for all $j \in J'$, and let $\hat{J} = \cup_{j \in J'} \hat{I}_j$. Also, $\pi_3 = 8\lambda + \pi_3'$, for some $\pi_3' \geq 0$. We show below that, for any $\lambda \in (0, \frac{1}{20})$, there exist constants π_1, π_2, π_3', such that (a) $4\lambda + 2(\pi_2 + 4\lambda) \leq 1 - \pi_3$ and (b) $\frac{4\pi_1}{\pi_2} + \pi_1 < 1$. Let $\lambda \in (0, \frac{1}{20})$, and select π_2, π_3', such that $20\lambda \leq 1 - \pi_3' - 2\pi_2$. Then, $12\lambda + 2(\pi_2 + 4\lambda) \leq 1 - \pi_3$, and (a) is satisfied. Now, setting $\pi_1 < \frac{1}{1+\frac{4}{\pi_2}}$, we have that (b) holds. To complete the proof, we show that the above implies that (i) all of the items in $\hat{J}$ can be packed into the empty bins in E, and (ii) the total expected utility is a constant fraction of the optimum. To show (i), we apply Theorem 3, setting $\hat{T} = E$, $\alpha = 1 - \pi_3$, $\beta = \pi_2 + 4\lambda$ and $\gamma = 4\lambda$. Then, from (a) we have that $\gamma + 2\beta \leq \alpha$. Also, we note that, by Lemma 7, for all $j \in \hat{J}$ such that $\chi = \mathfrak{L}(\chi_j)$, it holds that $|\hat{I}_j| \leq |I_j| \leq \lambda|\chi_j| \leq 4\lambda|\mathfrak{L}(\chi_j)| = \gamma|\chi|$ Thus, all of the conditions of Theorem 3 hold. Now, (ii) follows from (b). Hence, we obtain a constant approximation for any $\lambda \in (0, \frac{1}{20})$. ◀