The Generalized Assignment Problem with Minimum Quantities

Sven O. Krumke[a], Clemens Thielen[a,*]

[a] *University of Kaiserslautern, Department of Mathematics*
Paul-Ehrlich-Str. 14, D-67663 Kaiserslautern, Germany

Abstract

We consider a variant of the generalized assignment problem (GAP) where the amount of space used in each bin is restricted to be either zero (if the bin is not opened) or above a given lower bound (a *minimum quantity*). We provide several complexity results for different versions of the problem and give polynomial time exact algorithms and approximation algorithms for restricted cases. For the most general version of the problem, we show that it does not admit a polynomial time approximation algorithm (unless $\mathsf{P} = \mathsf{NP}$), even for the case of a single bin. This motivates to study *dual* approximation algorithms that compute solutions violating the bin capacities and minimum quantities by a constant factor. When the number of bins is fixed and the minimum quantity of each bin is at least a factor $\delta > 1$ larger than the largest size of an item in the bin, we show how to obtain a polynomial time dual approximation algorithm that computes a solution violating the minimum quantities and bin capacities by at most a factor $1 - \frac{1}{\delta}$ and $1 + \frac{1}{\delta}$, respectively, and whose profit is at least as large as the profit of the best solution that satisfies the minimum quantities and bin capacities strictly. In particular, for $\delta = 2$, we obtain a polynomial time $(1, 2)$-approximation algorithm.

Keywords: assignment problems, combinatorial optimization, approximation algorithms, computational complexity

1. Introduction

The generalized assignment problem (cf., for example, [1, 2]) is a classical generalization of both the (multiple) knapsack problem and the bin packing problem. In the classical version of GAP, one is given m bins, a capacity B_j for each bin j, and n items such that each item i has size $s_{i,j}$ and yields profit $p_{i,j}$ when packed into bin j. The goal is to find a feasible packing of the items into

*Corresponding author. Fax: +49 (631) 205-4737. Phone: +49 (631) 205-4590
Email addresses: krumke@mathematik.uni-kl.de (Sven O. Krumke), thielen@mathematik.uni-kl.de (Clemens Thielen)

the bins that maximizes the total profit. Applications of GAP include fixed charge location problems, grouping and loading for flexible manufacturing systems, vehicle routing, scheduling projects, allocating storage space, scheduling payments on accounts, designing communication networks, assigning software development tasks to programmers, assigning jobs to computers in networks, scheduling variable length TV commercials, and assigning ships to overhaul facilities. For details on these applications, we refer to [2] and the references therein.

In this paper, we consider a variation of the problem where the amount of space used in each bin is restricted to be either zero (if the bin is not opened) or above a given lower bound (a *minimum quantity*). This additional restriction is motivated from many practical packing problems where it does often not make sense to open an additional container (bin) if not at least a certain amount of space in it will be used. For example, in the classical application of GAP where the bins correspond to workers and the items correspond to jobs that can be assigned to them, it may be unreasonable to hire an additional worker if not at least a certain amount of work will be assigned to him.

Another motivation for adding minimum quantities to GAP is its application in unrelated machine scheduling [3]. Here, the bins correspond to machines and the items to jobs. In practice, it may be that running a machine is only feasible if a prescribed minimum load is attained on the machine. In a foundry or steel works, for example, a machine can only be used if it has a certain minimum amount of metal to process.

Formally, the generalized assignment problem with minimum quantities (GAP-MQ) is defined as follows:

Definition 1 (GAP with Minimum Quantities (GAP-MQ)).

INSTANCE:	m bins with capacities $B_1, \ldots, B_m \in \mathbb{N}$ and minimum quantities $q_1, \ldots, q_m \in \mathbb{N}$ (where $q_j \leq B_j$ for all $j = 1, \ldots, m$), and n items. For each item i and bin j, a size $s_{i,j} \in \mathbb{N}$ and a profit $p_{i,j} \in \mathbb{N}$.
TASK:	Find a packing of a subset of the items into a subset S of the bins such that the total space used in each bin $j \in S$ is at least q_j and at most B_j and the total profit is maximized.

In the decision version of the problem, a bound $P \in \mathbb{N}$ on the total profit is given and the question is whether there exists a feasible packing with total profit at least P. When stating results about the computational complexity of an optimization problem such as GAP-MQ, we will always mean the complexity of the corresponding decision problem.

Note that, in the above definition and throughout the paper, we always assume $\mathbb{N}$ to contain zero and denote the positive integers by $\mathbb{N}_+$. Moreover, we always assume that $q_j \leq \sum_{i=1}^{n} s_{i,j}$ for each bin j in an instance of GAP-MQ (otherwise, there does not exist any feasible solution that packs a nonempty set of items into bin j, so the bin can be removed from the instance).

Two important special cases of GAP-MQ motivated from the problem of assigning students to seminars at a university are the *seminar assignment problem* (SAP) and the *participant maximization problem* (PMP). SAP is the special case of GAP-MQ in which all item sizes are one. It can be motivated from the problem of assigning students (items) to seminars (bins) such that the number of participants in each seminar j is between q_j and B_j and the total satisfaction (profit) of the students is maximized. PMP is the special case of SAP in which all profits are one, so the objective is simply to maximize the number of students that receive a place in one of the seminars (the total number of participants). In these applications, the minimum quantities are a very natural restriction since holding a seminar only make sense if there is at least a certain minimum number of students giving a talk in the seminar.

By a *polynomial time α-approximation algorithm* for a maximization problem such as GAP-MQ, we mean an algorithm that, for any given instance I of encoding length $|I| \in \mathbb{N}_+$, finds a feasible solution with objective value at least $\frac{1}{\alpha}$ times optimal in time bounded by a polynomial in $|I|$ if the instance I admits a feasible solution, and outputs infeasibility of the instance after a number of steps bounded by a polynomial in $|I|$ if no feasible solution for instance I exists.

For problems which do not admit polynomial time approximation algorithms, a common approach is to study *dual* approximation algorithms that compute solutions violating some constraints of the problem by at most a constant factor. By a *polynomial time (α, β)-approximation algorithm* for a maximization problem, we mean an algorithm that, for any given instance I of encoding length $|I| \in \mathbb{N}_+$, achieves the following: If the instance I admits a feasible solution, the algorithm requires only time bounded by a polynomial in $|I|$ to find a solution that violates the constraints of the problem by at most a factor β and whose objective value is at least $\frac{1}{\alpha}$ times as large as the objective value of the optimal solution that satisfies the constraints strictly. If the instance I does not admit a feasible solution, the algorithm outputs infeasibility of the instance after a number of steps bounded by a polynomial in $|I|$.

1.1. Previous Work

The classical GAP is well-studied in literature. A comprehensive introduction to the problem can be found in [1]. A survey of algorithms for GAP is given in [2]. For a survey on different variants of assignment problems studied in literature, we refer to [4].

GAP is known to be APX-hard [5], but there exists a 2-approximation algorithm [3, 5]. Cohen et al. [6] showed how any polynomial time α-approximation algorithm for the knapsack problem can be translated into a polynomial time $(1+\alpha)$-approximation algorithm for GAP. A $(1, 2)$-approximation algorithm for the equivalent minimization version of GAP, assigning item i to bin j causes a cost $c_{i,j}$, was provided by Shmoys and Tardos [3]: For every feasible instance of GAP, their algorithm computes a solution that violates the bin capacities by at most a factor of 2 and whose cost is at most as large as the cost of the best solution that satisfies the bin capacities strictly.

GAP is a generalization of both the (multiple) knapsack problem (cf. [1, 5, 7]) and the bin packing problem (cf. [8, 9, 10]). The multiple knapsack problem is the special case of GAP where the size and profit of an item are independent of the bin (knapsack) it is packed into. The bin packing problem can be seen as the special case of the decision version of GAP in which all bins have the same capacity and all profits are one. The question of deciding whether a packing of total profit equal to the number of items exists is then equivalent to asking whether all items can be packed into the given number of bins.

A dual version of bin packing (often called *bin covering*) in which minimum quantities are involved was introduced in [11, 12]. Here, the problem is to pack a given set of items with sizes that do not depend on the bins so as to maximize the number of bins used, subject to the constraint that each bin contains items of total size at least a given threshold T (upper bin capacities are not considered due to the nature of the objective function). Hence, the bin covering problem can be seen as a variant of GAP-MQ in which the minimum quantity is the same for each bin and the objective is to maximize the number of bins used. Since any approximation algorithm with approximation ratio strictly smaller than 2 would have to solve the NP-complete partition problem when applied to instances in which the sizes of the items sum up to two, it follows that (unless $\mathsf{P} = \mathsf{NP}$) no polynomial time $(2-\epsilon)$-approximation for bin covering exists for any $\epsilon > 0$. In contrast, the main result of Assmann et al. [12] is an $\mathcal{O}(n \log^2 n)$ time algorithm that yields an *asymptotic* approximation ratio of 4/3 for bin covering, while easier algorithms based on next fit and first fit decreasing are shown to yield asymptotic approximation ratios of 2 and 3/2, respectively. Later, an asymptotic PTAS [13] and an asymptotic FPTAS [14] for bin covering were developed.

Minimum quantities have recently been studied for minimum cost network flow problems [15, 16, 17]. In this setting, minimum quantities for the flow on each arc are considered, which results in the minimum cost flow problem becoming strongly NP-complete [16]. Moreover, it was shown in [16] that (unless $\mathsf{P} = \mathsf{NP}$) no polynomial time $g(|I|)$-approximation for the problem exists for any polynomially computable function $g : \mathbb{N}_+ \to \mathbb{N}_+$, where $|I|$ denotes the encoding length of the given instance.

1.2. Our Contribution

We prove several complexity and approximation results on GAP-MQ and its special cases (see Table 1 for an overview).

We show that PMP is weakly NP-complete and admits a fully polynomial time approximation scheme (FPTAS). In contrast, we prove that SAP is strongly NP-complete and (unless $\mathsf{P} = \mathsf{NP}$) does not admit a polynomial time approximation scheme (PTAS) even if all profits are in $\{0, 1\}$. We show, however, that SAP can be solved in polynomial time by linear programming (or, more efficiently, by minimum cost flow computations) when the number of seminars (bins) is fixed. For the general case of GAP-MQ (which, by our results on SAP, is strongly NP-complete even if all item sizes are one), we show that the problem

also remains strongly NP-complete if all profits are one or if the profit obtained from packing an item into any bin equals the size of the item. Both results hold even for the case that the size of an item is independent of the bin it is packed into. Moreover, we prove that (unless $\mathsf{P} = \mathsf{NP}$) no polynomial time approximation algorithm exists for GAP-MQ even for these restricted profit values and only one bin. We show, however, that GAP-MQ can be solved optimally in polynomial time when the profit of an item is independent of the bin it is packed into and the maximum bin capacity $B_{\max}$ as well as the number of different item types $(s_1, \ldots, s_m, p)$ are fixed.

For the case that the number of bins is fixed, we present a pseudo-polynomial time dynamic programming algorithm for GAP-MQ. More importantly, when the number of bins is fixed and $q_j \geq \delta s_{i,j}$ for all i, j and some $\delta > 1$, we show how to obtain a polynomial time dual approximation algorithm that computes a solution violating the minimum quantities and bin capacities by at most a factor $1 - \frac{1}{\delta}$ and $1 + \frac{1}{\delta}$, respectively, and whose profit is at least as large as the profit of the best solution that satisfies the minimum quantities and bin capacities strictly. In particular, for $\delta = 2$, we obtain a polynomial time $(1, 2)$-approximation algorithm for the problem for the case that the number of bins is fixed and $q_j \geq 2s_{i,j}$ for all i, j. Moreover, we provide computational results, which show that, on average, this $(1, 2)$-approximation algorithm produces almost feasible solutions with superoptimal profit.

2. The Participant Maximization Problem

In this section, we consider the participant maximization problem (PMP), which is the special case of GAP-MQ in which all item sizes and profits are one. It can be thought of as the problem of assigning students to seminars at a university such that the number of participants in each seminar j is between q_j and B_j and the maximum possible number of students receive a place in one of the seminars. Formally, the problem is defined as follows:

Definition 2 (Participant Maximization Problem (PMP)).

INSTANCE:	The number n of students and m seminars with capacities $B_1, \ldots, B_m \in \mathbb{N}$ and minimum quantities $q_1, \ldots, q_m \in \mathbb{N}$ (where $q_j \leq B_j$ for all $j = 1, \ldots, m$).
TASK:	Find an assignment of a subset of the students to a subset S of the seminars such that the number of students in each seminar $j \in S$ is at least q_j and at most B_j and the total number of students assigned to seminars is maximized.

We start by showing that PMP is weakly NP-complete. To show NP-hardness, we use a reduction from the (weakly) NP-complete subset sum problem (cf. [18]). An instance of SUBSET SUM consists of $l + 1$ positive integers $a_1, \ldots, a_l, B \in \mathbb{N}$ and the question is whether there exists a subset S of $\{1, \ldots, l\}$ such that $\sum_{i \in S} a_i = B$.

	GAP-MQ	**SAP**	**PMP**
complexity results	strongly NP-complete and non-approximable (even for unit profits or profit = size)	strongly NP-complete and does not admit a PTAS (even for profits in $\{0, 1\}$)	weakly NP-complete
algorithmic results	polynomially solvable when profits independent of bins and max. bin capacity and #(item types) fixed; polynomial time $(1, 2)$-approximation and pseudo-polynomial time dynamic programming algorithm when #(bins) fixed	polynomially solvable when #(seminars) fixed	admits an FPTAS

Table 1: Summary of our results.

Theorem 1. *PMP is weakly NP-complete even if the capacity and the minimum quantity of each seminar are the same (i.e., $B_j = q_j$ for each $j = 1, \ldots, m$).*

Proof. Membership in NP is obvious as the number of students assigned to seminars in a given solution can be calculated easily in polynomial time. To show NP-hardness, we reduce SUBSET SUM to PMP in polynomial time. Given an instance $a_1, \ldots, a_l, B$ of SUBSET SUM, we construct an instance of PMP as follows: The number of students is $n := B$ and there are $m := l$ seminars. The capacity and minimum quantity of seminar j are $B_j := q_j := a_j$ for each $j = 1, \ldots, l$.

It is now easy to see that there exists a subset S of $\{1, \ldots, l\}$ such that $\sum_{i \in S} a_i = B$ if and only if all students can be assigned to seminars: Given such a subset S, we can assign exactly $a_i = B_i = q_i$ students to seminar i for each $i \in S$ to obtain an assignment of all students to seminars that respects the seminar capacities and minimum quantities. Conversely, given an assignment of all $n = B$ students to a subset $S \subseteq \{1, \ldots, l\}$ of the seminars, it follows that exactly $a_i = B_i = q_i$ students must be assigned to each seminar $i \in S$, so $\sum_{i \in S} a_i = B$. □

Note that the encoding length of an instance of PMP is polynomial in $\log n$, whereas encoding an instance of SAP or the general version of GAP-MQ requires a number of bits that is a polynomial in n. Thus, Theorem 1 has no direct implications for the complexity of these more general versions of the problem.

Even though solving PMP optimally is NP-complete, the following easy greedy algorithm obtains an approximation ratio of 2 for the problem: We first sort the seminars by nonincreasing minimum quantities, so that $q_1 \geq q_2 \geq \cdots \geq q_m$. Then we assign exactly q_j students to each seminar $j = 1, 2, \ldots, m$ until one of the following conditions is fulfilled:

1. All students have been assigned,
2. Less than q_l students remain unassigned after considering seminar $l - 1$,
3. Each seminar $j \in \{1, \ldots, m\}$ has been assigned exactly q_j students.

In Case 1, we have found an optimal assignment. In Case 3, we continue filling all seminars $j = 1, 2 \ldots, m$ to their capacities B_j until all students are assigned or all places in these seminars are filled, which again yields an optimal assignment. In Case 2, we must have $l \geq 2$, and the ordering of the q_j implies that

$$\frac{\mathsf{ALG}}{\mathsf{OPT}} \geq \frac{\sum_{j=1}^{l-1} q_j}{\sum_{j=1}^{l} q_j} \geq \frac{l-1}{l} \geq \frac{1}{2},$$

where OPT denotes the number of students assigned to seminars in an optimal assignment and ALG denotes the number of students assigned by the algorithm. Hence, the algorithm obtains an approximation ratio of 2 for PMP.

We now show how to obtain a fully polynomial time approximation scheme (FPTAS) for PMP. To do so, we will show how an arbitrary instance of PMP can

be solved by solving a suitable instance of the 0-1-knapsack problem (cf. [18]). Moreover, this reduction is polynomial time and preserves approximations, so it follows that we obtain an FPTAS for PMP by simply applying the well-known FPTAS for the knapsack problem (cf. [9]) to the corresponding knapsack instance.

An instance of the 0-1-knapsack problem (0-1-KP) consists of l objects with sizes $s_1, \ldots, s_l \in \mathbb{N}$ and profits $p_1, \ldots, p_l \in \mathbb{N}$, and a knapsack capacity $B \in \mathbb{N}$ and the task is to find a subset S of $\{1, \ldots, n\}$ such that $\sum_{i \in S} s_i \leq B$ and such that the total profit $\sum_{i \in S} p_i$ is maximized.

Now consider an arbitrary instance of PMP. Given the number n of students, the seminar capacities $B_1, \ldots, B_m$, and the minimum quantities $q_1, \ldots, q_m$, we define an instance of 0-1-KP as follows: There are $l := m$ objects. The size of object i is $s_i := q_i$ and its profit is $p_i := B_i$ for $i = 1, \ldots, m$. The knapsack capacity is $B := n$. Applying the FPTAS for 0-1-KP to this instance, we obtain, for any $\epsilon > 0$, a set $S \subseteq \{1, \ldots, n\}$ of objects such that

$$\sum_{i \in S} q_i = \sum_{i \in S} s_i \leq B = n \tag{1}$$

$$\sum_{i \in S} B_j = \sum_{i \in S} p_i \geq (1 - \epsilon) \cdot \mathsf{OPT}_{\mathsf{KP}}, \tag{2}$$

where $\mathsf{OPT}_{\mathsf{KP}}$ denotes the optimal profit possible for the knapsack instance. Moreover, any assignment of $k \leq n$ students to a set S of seminars directly yields a feasible packing of the knapsack with profit at least k: When packing exactly the objects in S, we obtain that

$$\sum_{i \in S} s_i = \sum_{i \in S} q_i \leq k \leq n = B \quad \text{and}$$

$$\sum_{i \in S} p_i = \sum_{i \in S} B_i \geq k,$$

where the first line holds since the assignment of students respects the minimum quantities and the second line holds since it respects the seminar capacities. Hence, it follows that $\mathsf{OPT}_{\mathsf{KP}} \geq \mathsf{OPT}$ holds for the optimal objective value OPT of the given instance of PMP, so (1) and (2) show that the set S obtained from the FPTAS for the knapsack instance yields a set of seminars that can all be filled to at least their minimum quantities (by (1)) and that provide enough places for at least $(1 - \epsilon) \cdot \mathsf{OPT}_{\mathsf{KP}} \geq (1 - \epsilon) \cdot \mathsf{OPT}$ students (by (2)). Filling each of the seminars in S to its minimum capacity and distributing the remaining students arbitrarily to the remaining places in the seminars in S, thus, yields a feasible solution for the given instance of PMP in which at least $(1 - \epsilon) \cdot \mathsf{OPT}$ students are assigned, so we have proved the following theorem:

Theorem 2. *There is an FPTAS for PMP.*

3. The Seminar Assignment Problem

In this section, we consider the seminar assignment problem (SAP), which is the special case of GAP-MQ in which all item sizes are one. It can be thought of as the problem of assigning students to seminars at a university such that the number of participants in each seminar j is between q_j and B_j and the and the total satisfaction (profit) of the students is maximized. Formally, the problem is defined as follows:

Definition 3 (Seminar Assignment Problem (SAP)).

INSTANCE:	The number n of students, m seminars with capacities $B_1, \ldots, B_m \in \mathbb{N}$ and minimum quantities $q_1, \ldots, q_m \in \mathbb{N}$ (where $q_j \leq B_j$ for all $j = 1, \ldots, m$), and a profit $p_{i,j} \in \mathbb{N}$ resulting from assigning student i to seminar j for $i = 1, \ldots, n$ and $j = 1, \ldots, m$.
TASK:	Find an assignment of a subset of the students to a subset S of the seminars such that the number of students in each seminar $j \in S$ is at least q_j and at most B_j and the total profit is maximized.

We start by showing that SAP does not admit a PTAS. More specifically, we show that there exists $\epsilon > 0$ such that it is strongly NP-hard to approximate SAP within a factor larger than $(1 - \epsilon)$. In particular, this implies strong NP-hardness of the problem. For the proof, we use a reduction from the 3-bounded 3-dimensional matching problem (3DM-3) defined as follows:

Definition 4 (3-Bounded 3-Dimensional Matching Problem (3DM-3)).

INSTANCE:	A set $T \subseteq X \times Y \times Z$, where $\lvert X\rvert = \lvert Y\rvert = \lvert Z\rvert = n$ and each element of $X \cup Y \cup Z$ appears at most three times as a coordinate of an element of T.
TASK:	Find a matching in T of maximum cardinality, i.e., a maximum cardinality subset $M \subseteq T$ such that no two elements of M agree in any coordinate.

3DM-3 is known to be APX-complete [19]. For our proof, we use a result of Petrank [20] who showed that 3DM-3 has a hard gap at location 1, i.e., there exists $\epsilon_0 > 0$ such that it is (strongly) NP-hard to decide whether a given instance of 3DM-3 has a matching of size n or if every matching has size at most $(1 - \epsilon_0)n$.

Theorem 3. *Let ϵ_0 be such that it is strongly NP-hard to decide whether a given instance of 3DM-3 has a matching of size n or if every matching has size at most $(1 - \epsilon_0)n$. Then it is strongly NP-hard to approximate SAP within a factor larger than $(1 - \frac{1}{3}\epsilon_0)$ even if the capacity and the minimum quantity of each seminar are fixed to 3 (i.e., $B_j = q_j = 3$ for each $j = 1, \ldots, m$) and all profits $p_{i,j}$ are in $\{0, 1\}$. In particular, SAP does not admit a PTAS.*

PROOF. Given an instance of 3DM-3 specified by $T \subseteq X \times Y \times Z$ with $|X| = |Y| = |Z| = n$ and $|T| = m$, we construct an instance of SAP as follows: There are $3n$ students corresponding to the $3n$ elements of $X \cup Y \cup Z$ and m seminars corresponding to the triples in T. The capacity and minimum quantity of each seminar are $B_j := q_j := 3$ for each $j = 1, \ldots, m$. The profit $p_{i,j}$ resulting from assigning student i to seminar j is one if element i is a coordinate of triple j and zero otherwise.

If the given instance of 3DM-3 has a matching of size n, we obtain an assignment of students to seminars with profit $3n$ by assigning exactly the students corresponding to the coordinates of each triple in the matching to the corresponding seminar. Conversely, if every matching has size at most $(1 - \epsilon_0)n$, any feasible assignment of students to seminars can contain at most $(1 - \epsilon_0)n$ seminars that yield a profit of 3. With the remaining $3n\epsilon_0$ students, at most $n\epsilon_0$ additional seminars can be opened, each of which can yield a profit of at most 2. Hence, the profit of any feasible solution is no larger than

$$3(1 - \epsilon_0)n + 2n\epsilon_0 = 3n(1 - \epsilon_0 + \frac{2}{3}\epsilon_0) = 3n(1 - \frac{1}{3}\epsilon_0),$$

which shows that it is NP-hard to approximate SAP within a factor larger than $(1 - \frac{1}{3}\epsilon_0)$. □

We now show how SAP can be solved optimally in polynomial time when the number m of seminars is fixed. To this end, first assume that the subset S of the seminars used in an optimal solution for a given instance of SAP is known. Given S, we can solve the given instance of SAP by solving the following integer linear program (ILP):

$$\max \quad \sum_{i=1}^{n} \sum_{j \in S} p_{i,j} x_{i,j} \tag{3}$$

$$\text{s.t.} \quad q_j \leq \sum_{i=1}^{n} x_{i,j} \leq B_j \qquad \forall\, j \in S \tag{4}$$

$$0 \leq \sum_{j \in S} x_{i,j} \leq 1 \qquad \forall\, i \in \{1, \ldots, n\} \tag{5}$$

$$0 \leq x_{i,j} \leq 1 \qquad \forall\, i \in \{1, \ldots, n\}, j \in S \tag{6}$$

$$x_{i,j} \in \mathbb{Z} \qquad \forall\, i \in \{1, \ldots, n\}, j \in S \tag{7}$$

It is now easy to see that the coefficient matrix of the variables $x_{i,j}$ in the inequalities (4) and (5) is totally unimodular. Hence, the Integrality-Theorem of Hoffman and Kruskal (cf. [21]) implies that the polyhedron defined by the inequalities (4) to (6) is integral, so we can solve the ILP by solving the corresponding linear program (LP), which can be done in polynomial time.

Since we assumed that the number m of seminars is fixed, we directly obtain a polynomial time algorithm for SAP in this case by solving the LP for every one of the at most 2^m possible choices of the set $S \subseteq \{1, \ldots, m\}$ of seminars to

use and then taking the solution obtaining the highest profit. Hence, we proved the following theorem:

Theorem 4. *SAP can be solved optimally in polynomial time when the number m of seminars is fixed.*

Note that, when requiring the linear program above to be solved in polynomial time, one needs to make use of a polynomial time algorithm for linear programming such as the ellipsoid method, which is rather inefficient in practice. We now show, however, that the subproblem of computing an optimal assignment for a given set S of seminars can be solved more efficiently by formulating it as a minimum cost flow problem. We consider the directed graph G shown in Figure 1. Here, the upper row of nodes corresponds to the students and the lower row corresponds to the seminars in the current test-set S and, for each student i and each seminar j, there is an arc (i, j) with cost $c = -p_{i,j}$ and upper capacity $u = 1$ from the node corresponding to i to the node corresponding to j. Additionally, there is a source node s that is connected by an arc (s, i) with cost $c = 0$ and upper capacity $u = 1$ to each node i corresponding to a student and a sink node t connected by an arc (j, t) with cost $c = 0$ and upper capacity $u = B_j$ and lower capacity $l = q_j$ to each node j corresponding to a seminar. Finally, there is an arc (s, t) with cost $c = 0$ and upper capacity $u = \infty$ connecting the source directly to the sink. All arcs except for the arcs (j, t) have lower capacity $l = 0$.

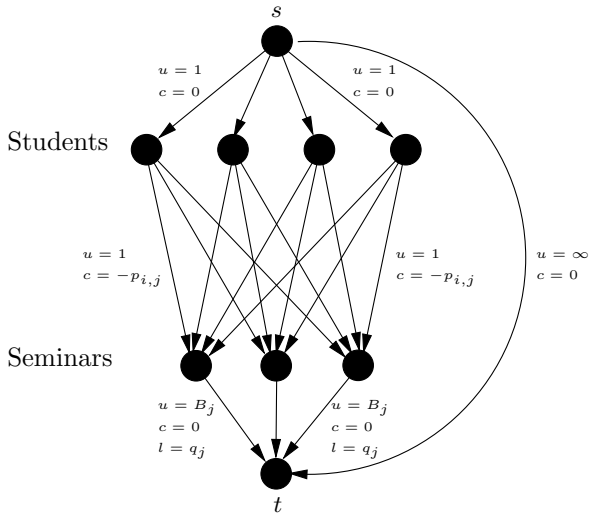

Figure 1: The graph G for $n = 4$ students and $m = 3$ seminars.

We now compute a minimum cost (s, t)-flow of flow value n in G (which can be done in strongly polynomial time, e.g., by the enhanced capacity scaling algorithm, cf. [22, 23, 24]). Since all capacities in G are integer, this minimum cost flow will be integer as well and, by construction of G, it corresponds exactly to an optimal assignment of students to seminars by assigning each student i

to the unique seminar j for which the flow on the arc (i, j) is 1 if such an arc exists, and leaving i unassigned otherwise. The number of unassigned students corresponds to the flow on the arc (s, t) and the cost of the flow equals minus the profit of the corresponding assignment. Hence, we can solve SAP by computing a minimum cost flow in G for every subset $S \subseteq \{1, \ldots, m\}$ and then taking the solution obtaining the highest profit.

4. GAP-MQ and GAP-MQ with Unit Profits

In this section, we consider the general case of GAP-MQ as well as the special case in which all profits are one and the item sizes do not depend on the bins. For the case of unit profits, we show that (just like SAP, the case of unit *sizes*) the problem is strongly NP-complete. Moreover, we show that it is weakly NP-hard to approximate the problem within any factor. In addition, we show that the same results also hold for the special case of GAP-MQ in which the profit obtained from packing an item into any bin equals the size of the item. In contrast, we show how the general version of GAP-MQ can be solved optimally in polynomial time when the profit of an item is independent of the bin it is packed into and the maximum bin capacity $B_{\max}$ as well as the number of different item types $(s_1, \ldots, s_m, p)$ are fixed. This approach generalizes methods which are well-known for the classical bin packing problem.

We start by proving NP-completeness of GAP-MQ with unit profits. To show NP-hardness, we use a reduction from the NP-complete problem 3-PARTITION (cf. [18]). An instance of 3-PARTITION consists of $l = 3k$ ($k \in \mathbb{N}$) positive integers $a_1, \ldots, a_l$ and a bound $L \in \mathbb{N}$ such that $L/4 < a_i < L/2$ for each i and $\sum_{i=1}^{l} a_i = kL$. The question is whether there exists a partition of $\{1, \ldots, l\}$ into k disjoint subsets $S_1, \ldots, S_k$ such that $\sum_{i \in S_j} a_i = L$ for each $j \in \{1, \ldots, k\}$ (which is only possible if each S_j contains exactly 3 elements).

Theorem 5. *GAP-MQ with unit profits (i.e., $p_{i,j} = 1$ for all i, j) is strongly NP-complete, even if the item sizes are independent of the bins (i.e., $s_{i,j} = s_i$ for all i, j).*

Proof. Membership in NP is obvious as the total profit of a given solution can be checked easily in polynomial time. To show NP-hardness, we reduce 3-PARTITION to GAP-MQ with unit profits. Given an instance $a_1, \ldots, a_l, L$ with $l = 3k$ of 3-PARTITION, we construct an instance of GAP-MQ as follows: There are $n := l$ items and the size of item i is $s_i := a_i$ for each $i \in \{1, \ldots, l\}$. There are k bins with $B_j := q_j := L$ for $j = 1, \ldots, k$. All profits $p_{i,j}$ are one. Then, since $L/4 < a_i < L/2$ for each i and $\sum_{i=1}^{l} a_i = kL$, it follows immediately that there exists a feasible packing of all items (which yields a total profit of n) into the k bins (where each bin must contain exactly 3 items) if and only if there exists a 3-partition of $a_1, \ldots, a_l$ (i.e., the given instance of 3-PARTITION is a yes-instance). □

Note that, when defining the profit $p_{i,j}$ obtained from packing item i into any bin j in the proof of Theorem 5 to be equal to the size s_i of item i, the same argumentation shows that there exists a 3-partition of $a_1, \ldots, a_l$ if and only if there exists a feasible packing (of all items) that yields a total profit of $\sum_{i=1}^{l} a_i = kL$. Hence, we obtain the following result on the complexity of the special case of GAP-MQ in which the profit obtained from packing an item into any bin equals the size of the item:

Corollary 6. *GAP-MQ remains strongly NP-complete if if the item sizes are independent of the bins and the profit obtained from packing an item into any bin equals the size of the item (i.e., $p_{i,j} = s_{i,j} = s_i$ for all i, j).*

We now consider the question of approximability of GAP-MQ with unit profits and show that approximating the problem within any factor is weakly NP-hard. For the proof, we use a reduction from SUBSET SUM (cf. Section 2).

Theorem 7. *Unless P = NP, there does not exists a polynomial time approximation algorithm for GAP-MQ with unit profits even for instances in which there is only one bin.*

Proof. Given an instance $a_1, \ldots, a_l, B$ of SUBSET SUM, we construct an instance of GAP-MQ as follows: There are l items with sizes $s_i := a_i$ and one bin with $B_1 := q_1 := B$. All profits are one. It is then clear that the given instance of SUBSET SUM has a solution if and only if there exists a solution of the GAP-MQ instance with positive profit. Hence, any approximation algorithm for GAP-MQ could be used to solve the weakly NP-complete subset sum problem. □

Again, the same argumentation can be used for the case that the profit obtained from packing an item equals its size, which yields the following corollary:

Corollary 8. *Unless P = NP, there does not exists a polynomial time approximation algorithm for GAP-MQ even for instances in which there is only one bin and the profit obtained from packing an item into the bin equals the size of the item.*

We now consider the general case of GAP-MQ and show how it can be solved optimally in polynomial time when the profit of an item is independent of the bin it is packed into and the maximum bin capacity $B_{\max}$ as well as the number of different item types $(s_1, \ldots, s_m, p)$ are fixed.

So assume that the profit of an item is independent of the bin it is packed into and the maximum bin capacity $B_{\max}$ as well as the number z of different item types $(s_1, \ldots, s_m, p)$ are fixed independently of the input size. For any given instance with m bins $(B_1, q_1), \ldots, (B_m, q_m)$, we let $N = \{(s_1^1, \ldots, s_m^1, p^1), \ldots, (s_1^z, \ldots, s_m^z, p^z)\}$ denote the set of different item types and let $n_1, \ldots, n_z$ be the numbers of items of each type.

Definition 5. A *(feasible) packing* of bin j is a vector $T = (T_1, \ldots, T_z)$ of non-negative integers such that $T_l \leq n_l$ for each $l \in \{1, \ldots, z\}$ and $q_j \leq \sum_{l=1}^{z} T_l s_j^l \leq B_j$.

Since each item size s_j^i is at least one, at most $B_{\max} = \max_{j=1,\dots,m} B^j$ items fit into any bin. Hence, since there are z possible item types, the number δ of possible packings satisfies $\delta \leq \frac{(z+1)^{B_{\max}}}{B_{\max}!}$ and a solution can be specified by providing the numbers $x_1, \dots, x_\delta$ of bins of each of the δ possible packings $T^1, \dots, T^\delta$ used. However, not every integral vector $(x_1, \dots, x_\delta)$ corresponds to a feasible solution of the problem. Denoting the set of bins for which packing T^t is feasible by $J(t)$, we need to make sure that the following conditions are satisfied:

$$\sum_{t=1}^{\delta} x_t T_l^t \leq n_l \quad \forall\, l \in \{1, \dots, z\} \tag{8}$$

$$\sum_{t \in S} x_t \leq |\cup_{t \in S} J(t)| \quad \forall\, S \subset \{1, \dots, \delta\} \tag{9}$$

$$x_t \geq 0 \quad \forall\, t \in \{1, \dots, \delta\} \tag{10}$$

Here, constraint (8) ensures that at most the available number n_l of items of the lth type are used. Constraint (9) ensures that every packing can be assigned to a bin it is feasible for such that no bin is assigned to more than one packing:

Lemma 9. *Given any integer solution $(x_t)_t$ to (9) and (10), it is possible to assign all packings T^t used by $(x_t)_t$ (where x_t determines the number of copies of packing T^t that are used) to the bins such that no bin is assigned to more than one packing.*

Proof. Consider the following bipartite graph $G = (U \dot\cup V, E)$: The set of vertices is $U \dot\cup V$, where $U = \{u_1, \dots, u_m\}$ corresponds to the m bins and $V = \cup_{t=1,\dots,\delta} \{v_1^t, \dots, v_{x_t}^t\}$ contains x_t vertices $v_1^t, \dots, v_{x_t}^t$ for each packing T^t. There are edges between all vertices $v_1^t, \dots, v_{x_t}^t$ and the vertex u_j corresponding to bin j if and only if packing T^t is feasible for bin j.

For every subset $S' \subseteq V$, we denote the set of vertices in U adjacent to the vertices in S' by $\mathrm{Adj}(S')$ and denote the set of packings to which the vertices in S' belong by $P(S')$. Since $(x_t)_t$ satisfies (9), we then have

$$|S'| \leq \sum_{t \in P(S')} x_t \leq |\cup_{t \in P(S')} J(t)| = |\mathrm{Adj}(S')| \quad \forall\, S' \subseteq V.$$

Hence, Hall's Theorem (cf. [25]) implies that there exists a matching of V into U in the bipartite graph G, i.e., a matching that matches each vertex in V to exactly one vertex in U. By construction of the graph G, this proves the claim. □

We can now formulate the given instance of GAP-MQ as the following integer

linear program (ILP):

$$\begin{aligned}
\max \quad & \sum_{t=1}^{\delta} \sum_{l=1}^{z} x_t T_l^t p^l & \\
\text{s.t.} \quad & \sum_{t=1}^{\delta} x_t T_l^t \leq n_l & \forall\, l \in \{1, \ldots, z\} \\
& \sum_{t \in S} x_t \leq |\cup_{t \in S} J(t)| & \forall\, S \subset \{1, \ldots, \delta\} \\
& x_t \in \mathbb{N} & \forall\, t \in \{1, \ldots, \delta\}
\end{aligned}$$

This ILP has δ variables and $z + 2^\delta + \delta$ constraints. Thus, as $\delta \leq \frac{(z+1)^{B_{\max}}}{B_{\max}!}$ with z and $B_{\max}$ fixed, the ILP has constant size and can be solved in time only depending on the encoding length of the numbers occurring in the ILP. Given an optimal solution $(x_t)_t$, Lemma 9 implies that there exists a matching of packings to bins, which can be computed in time $\mathcal{O}(m^{2.376})$ by applying a suitable matching algorithm (cf. [25]) to the bipartite graph G with at most $2m$ vertices defined in the proof of Lemma 9. Hence, we proved the following theorem:

Theorem 10. *GAP-MQ can be solved optimally in time $\mathcal{O}(m^{2.376})$ when the profit of an item is independent of the bin it is packed into and the maximum bin capacity $B_{\max}$ as well as the number of different item types $(s_1, \ldots, s_m, p)$ are fixed.*

5. GAP-MQ with Fixed Number of Bins

In this section, we present a pseudo-polynomial time dynamic programming algorithm and a polynomial time dual approximation algorithm for GAP-MQ in case that the number m of bins is fixed.

We start by deriving the dynamic programming algorithm. For a given instance of GAP-MQ, integers $P, S_1, \ldots, S_m \in \mathbb{N}$, and $i \in \{1, \ldots, m\}$, we define $T(P, S_1, \ldots, S_m, i)$ to be one if there exists a packing of the items $1, \ldots, i$ into the m bins that yields a profit of exactly P and packs items of size exactly S_j into bin j for each $j = 1, \ldots, m$. If such a packing does not exist, we define $T(P, S_1, \ldots, S_m, i)$ to be zero.

For $i = 1$, each value $T(P, S_1, \ldots, S_m, 1)$ can then be calculated in $\mathcal{O}(m)$ time as

$$T(P, S_1, \ldots, S_m, 1) = \begin{cases} 1 & \text{if } P = S_1 = \cdots = S_m = 0 \\ & \text{or } \exists j : P = p_{1,j}, S_j = s_{1,j}, S_k = 0 \text{ for } k \neq j \\ 0 & \text{else} \end{cases}$$

The values $T(P, S_1, \ldots, S_m, i)$ for $i > 1$ can then be computed recursively as

$$T(P, S_1, \ldots, S_m, i+1) = \max\Big\{T\big(P, S_1, \ldots, S_m, i\big), \max_{j=1,\ldots,m} T(P - p_{i+1,j}, S_1, \ldots, S_{j-1}, S_j - s_{i+1,j}, S_{j+1}, \ldots, S_m, i)\Big\},$$

so computing each value takes time $\mathcal{O}(m)$.

In order to obtain an optimal solution of the given instance of GAP-MQ, the algorithm computes the values $T(P, S_1, \ldots, S_m, i)$ for $0 \leq P \leq n \cdot p_{\max}$, $0 \leq S_j \leq B_j$ for $j = 1, \ldots, m$, and $1 \leq i \leq n$, where $p_{\max} := \max_{i,j} p_{i,j}$ denotes the maximum profit achievable from packing a single item. The optimum profit is then given as

$$\max\{0 \leq P \leq n \cdot p_{\max} : \exists S_1, \ldots, S_m \text{ with } T(P, S_1, \ldots, S_m, n) = 1 \text{ and } q_j \leq S_j \leq B_j \forall j\}$$

and the corresponding packing is an optimal solution of the instance.

Denoting the maximum bin capacity by $B_{\max} := \max_j B_j$, the total number of values $T(P, S_1, \ldots, S_m, i)$ computed in the algorithm can be upper bounded by $(n \cdot p_{\max} + 1) \cdot (B_{\max} + 1)^m \cdot n = \mathcal{O}(n^2 \cdot p_{\max} \cdot B_{\max}^m)$. Since each value can be computed in time $\mathcal{O}(m)$, the algorithm computes an optimum solution of the given instance of GAP-MQ in time $\mathcal{O}(n^2 \cdot m \cdot p_{\max} \cdot B_{\max}^m)$.

When the number of bins is fixed and $q_j \geq \delta s_{i,j}$ for all i, j and some $\delta > 1$, we now show how to obtain a polynomial time dual approximation algorithm that computes a solution violating the minimum quantities and bin capacities by at most a factor $1 - \frac{1}{\delta}$ and $1 + \frac{1}{\delta}$, respectively, and whose profit is at least as large as the profit of the best solution that satisfies the minimum quantities and bin capacities strictly. In particular, for $\delta = 2$, we obtain a polynomial time $(1, 2)$-approximation algorithm.

Our algorithm works by testing all 2^m possible subsets of bins that could be opened in a feasible solution, which are only a fixed number since m is assumed to be fixed. For a given subset $S \subseteq \{1, \ldots, m\}$ of bins, we then consider the following linear program:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} \sum_{j \in S} p_{i,j} x_{i,j} & & \\
\text{s.t.} \quad & \sum_{j \in S} x_{i,j} \leq 1 & & \forall\, i \in \{1, \ldots, n\} \\
& q_j \leq \sum_{i=1}^{n} s_{i,j} x_{i,j} \leq B_j & & \forall\, j \in S \\
& x_{i,j} \geq 0 & & \forall\, i \in \{1, \ldots, n\}, \\
& & & \forall\, j \in S
\end{aligned} \tag{LP(S)}$$

Note that adding the constraint $x_{i,j} \in \{0,1\}$ to (LP(S)) would yield a feasible integer programming formulation of the problem of finding a packing of maximum profit that uses exactly the bins in S. Hence, infeasibility of (LP(S)) implies that no feasible packing using exactly the bins in S exists. Our main result will be the following theorem:

Theorem 11. *Given a feasible solution x of (LP(S)) of profit P, we can compute in polynomial time an integer solution $\bar{x} \in \{0,1\}^{n \times |S|}$ of profit at least P for which $(1 - \frac{1}{\delta})q_j \leq \sum_{i=1}^{n} s_{i,j}\bar{x}_{i,j} \leq (1 + \frac{1}{\delta})B_j$ for each $j \in S$.*

Theorem 11 directly implies that we obtain a dual approximation for the given instance of GAP-MQ by computing an optimal solution of (LP(S)) for every possible subset $S \subseteq \{1, \ldots, m\}$ and then computing the corresponding integer solution for the fractional solution of highest profit. Hence, we obtain:

Theorem 12. *Let the number of bins be fixed and let $\delta > 1$ such that $q_j \geq \delta s_{i,j}$ for all i, j. Then there exists a polynomial time algorithm for the given instance of GAP-MQ that computes a solution violating the minimum quantities and bin capacities by at most a factor $1 - \frac{1}{\delta}$ and $1 + \frac{1}{\delta}$, respectively, and whose profit is at least as large as the profit of the best solution that satisfies the minimum quantities and bin capacities strictly.*

In particular, for $\delta = 2$, the solution computed by the algorithm from Theorem 12 violates the minimum quantities by at most a factor $\frac{1}{2}$ and the bin capacities by at most a factor $\frac{3}{2}$, so we obtain a $(1,2)$-approximation algorithm. This shows the following result:

Corollary 13. *There exists a $(1,2)$-approximation algorithm for GAP-MQ for the case that the number of bins is fixed and $q_j \geq 2s_{i,j}$ for all i, j.*

For the proof of Theorem 11, we need the following result about feasible solutions of (LP(S)):

Lemma 14. *If x is a feasible solution to (LP(S)), then $\sum_{i=1}^{n} x_{i,j} > 1$ for each $j \in S$.*

Proof. If $\sum_{i=1}^{n} x_{i,j} \leq 1$ for some $j \in S$, then

$$q_j \leq \sum_{i=1}^{n} s_{i,j}x_{i,j} \leq \max_{i=1,\ldots,n} s_{i,j} \cdot \sum_{i=1}^{n} x_{i,j} \leq \max_{i=1,\ldots,n} s_{i,j}.$$

As $\delta > 1$, this contradicts the assumption that $q_j \geq \delta s_{i,j}$ for all i, j. □

We now present the proof of Theorem 11 by providing an explicit algorithm that constructs the desired integer solution $\bar{x}$ from a given solution x. Our proof uses the same weighted bipartite graph $G(x)$ constructed from the given (fractional) solution x of (LP(S)) as in [3], where a $(1,2)$-approximation algorithm for the classical GAP was presented. We repeat the construction here for completeness. We note that our overall approach is similar to the one in [3],

but we need to modify the computation of the matching in the graph in order to ensure that the minimum quantities are only violated by certain factors.

In the bipartite graph $G(x) = (V, W, E)$, there is a vertex $w_i \in W$ for each item i, so we have $W = \{w_i : i = 1, \ldots, n\}$. For each bin $j \in S$, there are $k_j := \lceil \sum_{i=1}^{n} x_{i,j} \rceil$ vertices $v_{j,l}$, so $V = \{v_{j,l} : j = 1, \ldots, m, l = 1, \ldots, k_j\}$. Note that Lemma 14 implies that $k_j \geq 2$ for each $j \in S$.

The edges of $G(x)$ incident to the vertices $v_{j,l}$ corresponding to bin $j \in S$ and their weights x' are constructed as follows: We sort the items by nonincreasing size in bin j, so that $s_{1,j} \geq s_{2,j} \geq \cdots \geq s_{n,j}$. We then let $\hat{i}$ be the minimum index such that $\sum_{i=1}^{\hat{i}} x_{i,j} \geq 1$ (such an index exists by Lemma 14).

There is an edge $(v_{j,1}, w_i)$ for each $i \in \{1, \ldots, \hat{i} - 1\}$ for which $x_{i,j} > 0$, and for each of these edges, we set $x'(v_{j,1}, w_i) := x_{i,j}$. Moreover, there is an edge $(v_{j,1}, w_{\hat{i}})$ with $x'(v_{j,1}, w_{\hat{i}}) := 1 - \sum_{i=1}^{\hat{i}-1} x'(v_{j,1}, w_i)$ (so the sum of the components of x' for edges incident to $v_{j,1}$ is exactly 1).

If $\sum_{i=1}^{\hat{i}} x_{i,j} > 1$, a fraction of $x_{\hat{i},j}$ is still unassigned, so there is an edge $(v_{j,2}, w_{\hat{i}})$ with

$$x'(v_{j,2}, w_{\hat{i}}) := x_{\hat{i},j} - x'(v_{j,1}, w_{\hat{i}}) = \left(\sum_{i=1}^{\hat{i}} x_{i,j} \right) - 1.$$

We then continue with the items $i > \hat{i}$ of smaller size and construct edges $(v_{j,2}, w_i)$ until the sum of the components of x' for edges incident to $v_{j,2}$ have a weight of exactly 1, and so on. At the end, the definition of k_j implies that, for each of the vertices $v_{j,1}, \ldots, v_{j,k_j-1}$, the sum of the components of x' for edges incident to the vertex is exactly one, but this sum may be less than one for v_{j,k_j}.

It now follows that the vector x' on the edges of $G(x)$ is a *fractional matching*, i.e., all components are nonnegative and the sum of the components of x' on edges incident to each vertex is at most 1. Moreover, as we saw above, this fractional matching *exactly matches* each vertex $v_{j,l}$ for $l = 1, \ldots, k_j - 1$, i.e., for each of these vertices, the sum of the components of x' on edges incident to it is exactly 1. Moreover, if we define the *profit* of each edge $(v_{j,l}, w_i)$ to be $p_{i,j}$, the profit of x' is exactly equal to the profit P of x. Denoting, for each vertex $v_{j,l}$, the maximum and minimum size of an item i with $(v_{j,l}, w_i) \in E$ by $s_{j,l}^{\max}$ and $s_{j,l}^{\min}$, respectively, this shows the following lemma:

Lemma 15. *The vector x' is a fractional matching in $G(x)$ of profit at least P. It exactly matches each vertex $v_{j,l}$ for $j \in S$ and $l = 1, \ldots, k_j - 1$. Moreover, $s_{j,l}^{\min} \geq s_{j,l+1}^{\max}$ for $j \in S$ and $l = 1, \ldots, k_j - 1$.*

Our algorithm for constructing the desired integer solution $\bar{x}$ from x now works as follows:

Algorithm 1 (Constructing an Integer Solution $\bar{x}$ from a Fractional Solution x)**.**

Step 1: *Construct the bipartite graph $G(x)$ and the fractional matching x'.*
Step 2: *Compute a maximum profit (integer) matching M that exactly matches all vertices $v_{j,l}$ for $j \in S$ and $l = 1, \ldots, k_j - 1$.*
Step 3: *For each edge $(v_{j,l}, w_i) \in M$, assign item i to bin j, i.e., set $\bar{x}_{i,j} := 1$ if $(v_{j,l}, w_i) \in M$ and $\bar{x}_{i,j} := 0$, otherwise.*

We are now ready to prove Theorem 11. Here, the arguments used to show that the bin capacities will be violated by at most a factor of $(1 + \frac{1}{\delta})$ are similar to the ones used in the proof of the main result in [3]. Additionally, we need to ensure that the minimum quantities of all bins containing items are also violated by a factor of at most $(1 - \frac{1}{\delta})$. For this reason, we choose a matching that exactly matches all vertices $v_{j,l}$ for $j \in S$ and $l = 1, \ldots, k_j - 1$ in Step 2 of our algorithm, while the algorithm presented in [3] used a matching that exactly matches all the item vertices w_i in order to ensure that all items are packed (which is not required in our problem).

Proof of Theorem 11. We show that the integer solution $\bar{x}$ computed by Algorithm 1 is as desired. By Lemma 15, the vector x' is a fractional matching in $G(x)$ of profit at least P and matches all vertices $v_{j,l}$ for $j \in S$ and $l = 1, \ldots, k_j - 1$ exactly. The polytope of all fractional matchings in $G(x)$ is defined by the following constraints:

$$\begin{aligned} \sum_{i=1}^{n} y(v_{j,l}, w_i) &\leq 1 && \forall j \in S, l = 1, \ldots, k_j, \\ \sum_{j \in S, l=1,\ldots,k_j} y(v_{j,l}, w_i) &\leq 1 && \forall i = 1, \ldots, n, \\ y(v_{j,l}, w_i) &\geq 0 && \forall j \in S, l = 1, \ldots, k_j, i = 1, \ldots, n \end{aligned}$$

Writing these constraints as $Ay \leq 1, y \geq 0$, the corresponding matrix A is totally unimodular (cf., for example, [8]). Moreover, the requirement that the vertices $v_{j,l}$ for $j \in S$ and $l = 1, \ldots, k_j - 1$ are matched exactly only means adding the constraint $-\sum_{i=1}^{n} y(v_{j,l}, w_i) \leq -1$ for each such $v_{j,l}$, and the coefficient vector of this constraint is exactly the negative of the coefficient vector of the constraint $\sum_{i=1}^{n} y(v_{j,l}, w_i) \leq 1$. Hence, the matrix including these additional constraints is also totally unimodular (cf., for example, [8]), so the polytope of all fractional matchings that exactly match the vertices $v_{j,l}$ for $j \in S$ and $l = 1, \ldots, k_j - 1$ is integral. Consequently, we can compute a maximum profit (integer) matching in Step 2 of the algorithm in polynomial time and this matching has profit at least P. Since the integer solution $\bar{x}$ has exactly the same profit, this shows that $\bar{x}$ has profit at least P.

We now consider the total size of the items assigned to bin j for each $j = 1, \ldots, m$ and first show that this size is at most $(1 + \frac{1}{\delta})B_j$. There are k_j vertices $v_{j,l}$ corresponding to bin j in $G(x)$ and, for each of these, at most one item corresponding to an incident edge will be assigned to bin j. Hence, the total

size of the items assigned to bin j is at most $\sum_{l=1}^{k_j} s_{j,l}^{\max}$. Since $B_j \geq q_j \geq \delta s_{i,j}$ for all i, we have $s_{j,1}^{\max} \leq \frac{1}{\delta} B_j$. Moreover, Lemma 15 and the construction of x' imply that

$$\sum_{l=2}^{k_j} s_{j,l}^{\max} \leq \sum_{l=1}^{k_j-1} s_{j,l}^{\min} \leq \sum_{l=1}^{k_j-1} \sum_{i:(v_{j,l},w_i)\in E} s_{i,j} x'(v_{j,l}, w_i)$$
$$\leq \sum_{l=1}^{k_j} \sum_{i:(v_{j,l},w_i)\in E} s_{i,j} x'(v_{j,l}, w_i) = \sum_{i=1}^{n} s_{i,j} x_{i,j} \leq B_j,$$

where the last inequality holds since x is a feasible solution for (LP(S)). Consequently, we obtain that the total size of the items assigned to bin j is at most $(1 + \frac{1}{\delta})B_j$.

It remains to show that the total size of the items assigned to bin j is at least $(1 - \frac{1}{\delta})q_j$. Since the matching M computed in Step 2 of the algorithm matches all vertices $v_{j,l}$ for $l = 1, \ldots, k_j - 1$ exactly, the total size of the items assigned to bin j is at least $\sum_{l=1}^{k_j-1} s_{j,l}^{\min}$ and, by Lemma 15,

$$\sum_{l=1}^{k_j-1} s_{j,l}^{\min} \geq \sum_{l=2}^{k_j} s_{j,l}^{\max} = \sum_{l=1}^{k_j} s_{j,l}^{\max} - s_{j,1}^{\max} \geq \sum_{l=1}^{k_j} \sum_{i:(v_{j,l},w_i)\in E} s_{j,l}^{\max} x'(v_{j,l}, w_i) - s_{j,1}^{\max}$$
$$\geq \sum_{l=1}^{k_j} \sum_{i:(v_{j,l},w_i)\in E} s_{i,j} x'(v_{j,l}, w_i) - s_{j,1}^{\max} = \sum_{i=1}^{n} s_{i,j} x_{i,j} - s_{j,1}^{\max}$$
$$\geq q_j - s_{j,1}^{\max} \geq q_j - \frac{1}{\delta} q_j = (1 - \frac{1}{\delta}) q_j,$$

where we used that $q_j \geq \delta s_{j,1}^{\max}$. This shows that the total size of the items assigned to bin j is at least $(1 - \frac{1}{\delta})q_j$, which finishes the proof. □

6. Computational Results

In this section, we present computational results that provide insights into the average quality of the solutions obtained by our dual approximation algorithm presented in Section 5. For our tests, we choose instances for which $q_j \geq 2s_{i,j}$ for all i, j, so by choosing $\delta := 2$ our algorithm is guaranteed to provide a $(1, 2)$-approximation on these instances.

Note that, since the algorithm has to solve the linear program (LP(S)) for all 2^m possible subsets of bins that could be opened, the running time is of course prohibitive for large values of m. Hence, we tested only instances with at most 10 bins. The running time of our algorithm on each of these instances was less than 5 minutes on a standard desktop computer equipped with an Intel Core 2 Quad Q6700 CPU (2.66 GHz) and 8 gigabytes of memory.

In order to compute the optimum profit of a solution that respects all bin capacities and minimum quantities, we use the natural integer programming formulation of the problem:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} p_{i,j}x_{i,j} & \\
\text{s.t.} \quad & \sum_{j=1}^{m} x_{i,j} \leq 1 & \forall\, i \in \{1,\ldots,n\} \\
& q_j y_j \leq \sum_{i=1}^{n} s_i x_{i,j} \leq B_j y_j & \forall\, j \in \{1,\ldots,m\} \\
& x_{i,j}, y_j \in \{0,1\} & \forall\, i \in \{1,\ldots,n\}, \\
& & \forall\, j \in \{1,\ldots,m\}
\end{aligned}$$

Here, variable $x_{i,j}$ is one if and only if item i is packed into bin j and variable y_j is one if and only if bin j is opened.

The random instances in our computational experiments are generated as follows: The size of each item in each bin is chosen uniformly at random in $\{1,\ldots,20\}$. For each bin j, we then choose the bin capacity B_j uniformly at random in $\{2\cdot s^{\max}(j),\ldots,2\cdot s^{\max}(j)+50\}$, where $s^{\max}(j)$ denotes the maximum size of an item in bin j. The minimum quantity q_j of bin j is chosen uniformly at random in $\{2\cdot s^{\max}(j)+25,\ldots,2\cdot s^{\max}(j)+50\}$, where the values of B_j and q_j are exchanged in case that $q_j > B_j$. For the profits, we choose a profit parameter p_j uniformly at random from $\{0,\ldots,100\}$ for each bin j. The profit $p_{i,j}$ obtained from packing item i into bin j is then chosen uniformly at random from $\{0,\ldots,p_j\}$ (hence, p_j can be interpreted as the "attractiveness" of bin j).

For every combination of $n \in \{10,20,30,40,50\}$ and $m \in \{5,7,10\}$, we sampled 100 instances in this way and solved the integer program obtained from the formulation above using CPLEX 12.4. Our dual approximation algorithm was implemented as an OPL model in IBM ILOG CPLEX Optimization Studio 12.4. An optimal solution of the linear program (LP(S)) for each possible subset S of bins and the maximum profit matching M in Step 2 of Algorithm 1 were computed using CPLEX 12.4.

Table 2 shows the results of our computational experiments. For each instance size, the table shows the average values of the optimum profit of a solution that respects all bin capacities and minimum quantities (calculated by solving the integer program), the profit obtained by the dual approximation algorithm, the number of bins with violated minimum quantity, and the number of bins with violated bin capacity. Moreover, the last two columns provide the average factor by which the minimum quantity or the bin capacity are violated, each averaged over the violated values only.

n	m	average optimum profit	average profit	average # q_j violated	average # B_j violated	average q_j violation	average B_j violation
10	5	471.31	504.01	0.96	0.23	0.87	1.07
20	5	996.35	1023.15	1.07	1.09	0.88	1.09
30	5	1476.83	1514.89	0.95	1.83	0.88	1.10
40	5	1882.67	1934.72	0.68	2.41	0.89	1.10
50	5	2119.95	2175.68	0.53	2.70	0.92	1.09
10	7	484.4	515.10	0.90	0.25	0.86	1.09
20	7	1122.83	1147.56	1.15	0.92	0.89	1.10
30	7	1656.65	1688.9	1.18	1.67	0.89	1.09
40	7	2154.71	2201.17	1.00	2.43	0.90	1.09
50	7	2547.76	2596.52	0.94	2.86	0.90	1.09
10	10	540.85	569.74	0.92	0.19	0.87	1.10
20	10	1238.03	1266.28	1.37	0.69	0.87	1.10
30	10	1870.65	1900.90	1.39	1.58	0.87	1.09
40	10	2443.44	2481.04	1.19	2.29	0.89	1.10
50	10	2995.02	3043.72	1.34	3.00	0.90	1.09

Table 2: Computational results.

As the results show, our dual approximation algorithm produces solutions in which the minimum quantity is violated only for a very small number of bins on average. The bin capacity is violated slightly more often for the instances with larger numbers of items, but it is still respected for most bins. More importantly, the results show that the minimum quantities and bin capacities of the violated bins are only violated by about 9-14% on average for every instance size tested. Hence, the algorithm produces solutions with superoptimal profit that violate the constraints of the problem only by a small factor.

7. Future Research

An interesting theoretical question is whether there exist good polynomial time approximation algorithms for SAP or whether the problem is inapproximable (which we proved for GAP-MQ). We suspect that such approximation algorithms exist, but achieving a constant factor may be hard as most standard techniques for designing approximation algorithms fail for this problem.

References

[1] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Chichester, 1990.

[2] D. G. Cattrysse, L. N. Van Wassenhove, A survey of algorithms for the generalized assignment problem, European Journal of Operational Research 60 (3) (1992) 260–272.

[3] D. B. Shmoys, É. Tardos, An approximation algorithm for the generalized assignment problem, Mathematical Programming 62 (1993) 461–474.

[4] D. W. Pentico, Assignment problems: A golden anniversary survey, European Journal of Operational Research 176 (2) (2007) 774–793.

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

[6] R. Cohen, L. Katzir, D. Raz, An efficient approximation algorithm for the generalized assignment problem, Information Processing Letters 100 (4) (2006) 162–166.

[7] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer, 2004.

[8] G. L. Nemhauser, L. A. Wolsey, Integer and Combinatorial Optimization, Wiley-Interscience, New York, 1988.

[9] V. V. Vazirani, Approximation Algorithms, Springer, 2001.

[10] E. G. Coffman Jr., M. R. Garey, D. S. Johnson, Approximation algorithms for bin packing: A survey, in: Hochbaum [26], pp. 46–93.

[11] S. F. Assmann, Problems in discrete applied mathematics, Ph.D. thesis, Massachusetts Institute of Technology (1983).

[12] S. F. Assmann, D. S. Johnson, D. J. Kleinman, J. Y.-T. Leung, On a dual version of the one-dimensional bin packing problem, Journal of Algorithms 5 (4) (1984) 502–525.

[13] J. Csirik, D. S. Johnson, C. Kenyon, Better approximation algorithms for bin covering, in: Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA), 2001, pp. 557–566.

[14] K. Jansen, R. Solis-Oba, An asymptotic fully polynomial time approximation scheme for bin covering, Theoretical Computer Science 306 (2003) 543–551.

[15] H. G. Seedig, Network flow optimization with minimum quantities, in: Operations Research Proceedings 2010: Selected Papers of the Annual International Conference of the German Operations Research Society, Springer, 2011, pp. 295–300.

[16] S. O. Krumke, C. Thielen, Minimum cost flows with minimum quantities, Information Processing Letters 111 (11) (2011) 533–537.

[17] X. Zhu, Q. Yuan, A. Garcia-Diaz, L. Dong, Minimal-cost network flow problems with variable lower bounds on arc flows, Computers and Operations Research 38 (8) (2011) 1210–1218.

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

[19] V. Kann, Maximum bounded 3-dimensional matching is MAX SNP-complete, Information Processing Letters 37 (1) (1991) 27–35.

[20] E. Petrank, The hardness of approximation: Gap location, Computational Complexity 4 (2) (1994) 133–157.

[21] A. J. Hoffman, J. B. Kruskal, Integral boundary points of convex polyhedra, in: Kuhn and Tucker [27], pp. 223–246.

[22] J. B. Orlin, A faster strongly polynomial minimum cost flow algorithm, in: Proceedings of the 20th ACM Symposium on the Theory of Computing (STOC), 1988, pp. 377–387.

[23] S. Plotkin, É. Tardos, Improved dual network simplex, in: Proceedings of the 1st ACM-SIAM Symposium on Discrete Algorithms (SODA), 1990, pp. 367–376.

[24] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network Flows, Prentice Hall, 1993.

[25] A. Schrijver, Combinatorial Optimization, Vol. 24 of Algorithms and Combinatorics, Springer, 2003.

[26] D. S. Hochbaum (Ed.), Approximation Algorithms for NP-Hard Problems, PWS Publishing, Boston, 1997.

[27] H. W. Kuhn, A. J. Tucker (Eds.), Linear Inequalities and Related Systems, Princeton University Press, 1956.