The Open University of Israel
Department of Mathematics and Computer Science

Approximation Algorithms for Generalized Assignment Problems

Thesis submitted as partial fulfillment of the requirements
towards an M.Sc. degree in Computer Science
The Open University of Israel
Computer Science Division

By
Israel Beniaminy

Prepared under the supervision of Dr. Zeev Nutov

June 2005

Abstract

We consider a class of max-profit scheduling problems that occur naturally in many different applications, all involving assignment of jobs to multiple resources under a set of constraints. In the *Max-Profit Generalized Assignment Problem* (Max-GAP), we are given a set J of m bins (knapsacks), and a set I of n items. Each bin $j \in J$ has capacity $c(j)$. Each item $i \in I$ has in bin j size $\ell(i,j)$ and profit $p(i,j)$. The objective is to find a maximum profit feasible assignment. The problem admits a 1/2-approximation algorithm. Our main result is a $(1-1/e)$-approximation algorithm for *Max-GAP with fixed profits* when each item i has a fixed profit $p(i,j) = p(i)$ in every bin j. A particular case of Max-GAP with fixed profits is the *Multiple Knapsack with Assignment Restrictions* (MKAR) problem, where each bin $j \in J$ has capacity $c(j)$ and a specified set $I(j)$ of items that can be assigned to it, and each item i has size $\ell(i)$ and profit $p(i)$. We show that this version is APX-hard, and give a fast 1/2-approximation algorithm.

Acknowledgements

I wish to thank my thesis advisor, Dr. Zeev Nutov, for his friendly, knowledgeable and invaluable guidance, advice and support throughout the gestation and development of the results presented in this thesis. He played an important role in my effort to take the path of rigorous mathematical approach to the problems discussed here, rather than the heuristic approaches I have developed in my commercial work, and thus gain an insight towards how these two disciplines inform each other. Dr. Nutov's patience, encouragement, inspiration and time investment in guiding me through this process are all much appreciated.

Thanks to Dr. Raphael Yuster for showing how to derandomize the algorithm of our main result, and for tightening the analysis of the algorithm's approximation ratio. Thanks to Dr. Guy Kortsarz for valuable discussions, comments and ideas. I also thank an anonymous referee for useful comments on a draft paper presenting our key findings.

I also thank my colleagues at ClickSoftware Technologies for stimulating discussions and ideas.

I am thankful to my parents, who taught me that learning is a lifetime activity, and that it is never too late to look for new types of knowledge. It was in this spirit that I started my M.Sc. studies long after establishing a commercial career. I was looking forward to showing my mother the end-result of these studies. Instead, completing this thesis is yet another painful reminder that she is no longer with us. I miss her terribly.

My deepest thanks go to my son Gilad, my daughter Shira, and my wife Nehama for their love and understanding through the long process of studying towards the M.Sc. degree and of research towards this thesis. Nehama, without your encouragement, patience, shared joy, support and ever-present love, this thesis would not have been possible. You have facilitated my work in so many ways, taking far more than your share of managing our children and household, supporting me through tough times, gently prodding me when I needed it, and always believing that I could finish this work even when I almost didn't. Thanks for being with me on this journey - I am indebted to you for this work, and for everything else, more than I could ever say. I gratefully and whole-heartedly dedicate this thesis to you.

Contents

1 Introduction

We consider a class of problems that occur naturally in many different applications, all requiring the profit-maximization of assignment of jobs to multiple resources under a set of constraints. Such applications include inventory matching [4], loading containers [8], multiprocessor scheduling [12] and storage management in multimedia systems [13].

As most variants of such problems are NP-Complete, it is interesting to investigate polynomial-time algorithms that compute approximate solutions. Such algorithms are characterized by their **approximation ratio**, defined as follows:

Definition 1.1 *A* **ρ-approximation algorithm** *($\rho < 1$) for a maximization problem is a polynomial-time algorithm that computes a solution of value at least ρ times the optimal value. ρ is called an approximation ratio for the algorithm.*

In this work, we consider contant-ratio approximation algorithms - that is, algorithms whose approximation ratio does not depend on problem size.

This work is organized as follows: Section 2 defines the problems under consideration, highlights the relationships between them, and reviews previously published results. Section 3 presents detailed proofs for the best known previous results for approximating GAP (the General Assignment Problem) - a 1/2-approximation due to Chekuri and Khanna [3], based on a result by Shmoys and Tardos [11]. Section 4 presents our main result: a better approximation for an important restriction of GAP - Fixed-Profit GAP. Section 5 discusses MKAR, giving approximation hardness results and a fast combinatorial 1/2-approximation algorithm for GAP.

2 Problems Considered in this Research

A substantial part of the existing literature on scheduling addresses *minimization* goals, such as mimimizing the "makespan" - the time until the last job is completed. This work addresses *maximization* goals, involving some capacity constraints such as container size or available time. With such problems, the goals are typically maximizing the total profit of the jobs or items that can be scheduled or placed, given the stated capacity constraints. Such maximization is often a more accurate reflection of real-life applications, while still admitting simple and concise problem statements.

Throughout this work we assume without loss of generality that all the problem parameters are integers.

2.1 The Generalized Assignment Problem (GAP)

We consider the following problem:

Generalized Assignment Problem (GAP):
Instance: A set J of m bins (knapsacks), and a set I of n items. Each bin $j \in J$ has a capacity $c(j)$. For each item i and bin j we have size $\ell(i,j)$ and profit $p(i,j)$.
Objective: Find a maximum profit feasible assignment.

To be more precise, an assignment for GAP where every item is assigned to at most one bin is feasible if the capacity constrains are satisfied; namely, $\ell(A_j) = \sum_{i \in A_j} \ell(i,j) \leq c(j)$ for every bin $j \in J$ and the set A_j of items assigned to it.

Remark: GAP has also been defined in the literature as a closely related minimization problem. In this thesis, following [3], we use the term **GAP** to refer to the maximization version of the problem. We will refer to the minimization version as **Min-GAP**. The minimization version replaces the profits $p(i,j)$ with costs $w(i,j)$. In Min-GAP, the goal is to find a feasible assignment for all of the items, such that the total cost for the assigned items is minimized. Some research on Min-GAP, such as [11], has also addressed minimizing the *makespan* - the time until the last job is completed.

Fixed-Profit GAP is a restriction of GAP where $p(i,j)$ is fixed for all $j \in J$, and may therefore be written as $p(i)$.

2.2 Multi-Knapsack with Assignment Restrictions (MKAR)

MKAR is a special case of GAP, where the item size and its profit do not depend on the bin it is assigned to:

Multi-Knapsack with Assignment Restrictions (MKAR):
Instance: A set J of m bins (knapsacks), and a set I of n items. Each bin $j \in J$ has capacity $c(j)$ and a specified set $I(j)$ of items that can be assigned to it. Each item $i \in I$ has size $\ell(i)$ and profit $p(i)$.
Objective: Find a maximum-profit feasible assignment.

2.3 Related problems

MKAR is a generalization of the extensively researched **Multiple Knapsack Problem** (MKP, [3]). In MKP, any item can be assigned to any bin, provided only that the item's size does not exceed the bin's capacity. In the terms of MKAR's definition, $I(j) = I$, and thus there are no assignment restrictions.

An extension of MKAR that occurs in some natural applications is the **Multi-Knapsack problem with Color constraints Problem** [4].

Definition 2.1 MKARCC*, or Multi-Knapsack with Assignment Restrictions and Color Constraints Problem, is a generalization of the multi-knapsack problem. In this generalization, a set of colors C is defined, and each item is associated with a color $c(i) \in C$. The* color constraints *require that the number of distinct colors for items assigned to any specific knapsack must be less than a constant K.*

A related problem is discussed in [10] under the name **Class-Constrained Multiple Knapsack Problem** (CCMK).

2.4 Previous work

A 1/2-approximation algorithm for Max-GAP was given by Chekuri and Khanna [3] based on a result of Shmoys and Tardos [11], that considered the corresponding minimization version Min-GAP: instead of profits $p(i,j)$ there are costs $w(i,j)$, and the objective is to find a feasible assignment of all items (assuming such exists) while minimizing the total cost.

Definition 2.2 *A* pseudopacking *is an assignment so that $\ell(i,j) \leq c(j)$ for each item i assigned to bin j, and that can be made feasible by removing from each bin at most one item.*

Since checking whether all items can be assigned is an NP-hard problem, Shmoys and Tardos [11] proved that for Min-GAP there is a polynomial-time algorithm that finds a pseudopacking of cost $\leq$ *opt*. Using this, Chekuri and Khanna [3] proved that for GAP (the maximization version), there exists a polynomial-time algorithm that returns a pseudopacking of profit $\geq$ *opt*. This implies that GAP admits a 1/2-approximation algorithm (see [3]). Chekuri and Khanna also showed that GAP is APX-hard even for some simple instances.

A particular well-studied case of MKAR is the Multi-Knapsack Problem, with $I(j) = I$ for all j, for which Chekuri and Khanna [3] gave a PTAS. The difficulty of MKAR compared to Multi-Knapsack is in assigning items to "correct" bins. Another particular case is when $p(i) = \ell(i)$ for all i; for this case several 1/2-approximation algorithms are given in [5].

We are not aware of any previously published approximation results for MKARCC. For the related CCMK problem, [10] gives a PTAS. Since we show in this work that MKAR is APX-hard, and any instance of MKAR is easily reduced to an instance of MKARCC where all colors are different, it is unlikely that a PTAS exists for MKARCC. The key difference between CCMK and MKARCC are the assignment restrictions in MKARCC.

2.5 This work: New results for GAP and MKAR

Our main result is a $(1-1/e)$-approximation algorithm for fixed-profit GAP. As stated above, fixed-profit GAP may be viewed as a generalization of MKAR. We also extend this result to MKARCC, which is another generalization of MKAR.

We also show that MKAR is APX-hard, and give a combinatorial, simple and fast 1/2-approximation algorithm, generalizing [5].

3 A $1/2$-Approximation Algorithm for GAP

This section presents results due to Shmoys and Tardos [11], and to Chekuri and Khanna [3]. We have made minor enhancements to the discussion in order to consider bin capacities (which are not considered in the main result of [11]), and in order to provide more detail for the definition and proof of the bi-criteria approximation.

3.1 Min-GAP and Psuedo-Packings

We represent a Min-GAP instance by the the following linear program $LP(C, T)$. Integer solutions to this program have a one-to-one correspondence with schedules of cost at most C and makespan at most T, where all items are scheduled. Note that this program does not state any minimization or maximization objective, as we are concerned at this point only with finding feasible solutions obeying the constraints on C and T.

$$\begin{array}{ll}
\sum_{i\in I}\sum_{j\in J} w(i,j)x_{ij} \le C & \\
\sum_{j\in J} x_{ij} = 1 & \forall i \in I \\
\sum_{i\in I} \ell(i,j)x_{ij} \le \min(T, c(j)) & \forall j \in J \\
x_{i,j} \ge 0 & \forall i \in I, j \in J \\
x_{i,j} = 0 & \forall i \in I, j \in J \text{ so that } \ell(i,j) > \min(T, c(j))
\end{array} \tag{1}$$

Let $x = \{x_{ij}\}$ be a feasible solution to (1). Note that x_{ij} may be interpreted as the fraction of item i assigned to bin j. The first constraint guarantees total cost less than C. The second constraint requires that all items are scheduled. The third constraint requires that the total size of items assigned to a bin does not exceed its capacity, and does not exceed the maximum allowed makespan. The fourth constraint enforces non-negative solutions, and the fifth guarantees that no item is assigned if it is too large for its bin or for the specified makespan: such items cannot participate in feasible integer solutions.

Shmoys and Tardos [11] proved:

Theorem 3.1 ([11]) *There exists a polynomial-time algorithm that, given a feasible (possibly fractional) solution to $LP(C, T)$, generates a pseudopacking for the Min-GAP instance represented by $LP(C, T)$. This pseudopacking is an assignment with cost of at most C, a makespan of at most $2T$, and, for any bin that is overpacked (i.e., the total size of assigned items exceeds either bin size or T), there exists one item whose removal would result in a* *feasible packing for that bin.*

The following algorithm is used to prove this theorem: We use the fractional solution of (1) to construct a bipartite graph $B = (V, U, E)$, and assign a value $x'(v, u)$ to each edge $(v, u) \in E$. One side of B consists of node set U representing the items, where node u_l corresponds to item i_l for $1 \leq l \leq n$. The other side of B consists of node set $V = \{v_{js} : j = 1, \ldots, m, s = 1, \ldots, k_j\}$, where $k_j = \lceil \sum_i x_{ij} \rceil$. Thus, a bin j corresponds to a set of nodes in V, labeled v_{js}.

For each $(item, bin)$ pair with $x_{ij} \geq 0$, we add one or more edges to B. Each of these edges is assigned the cost $w(i, j)$. The algorithm uses two main variables: The variable s is the index of the current node in V out of the set of nodes corresponding to the bin being processed. The variable *currtotal* records the total of x' over the edges in B incident on the currently processed node (v_{js}). We also extend a vector x' to the edges in B, as follows:

```
For each bin $j$ do:
    $currtotal \leftarrow 0, s \leftarrow 1$
    For each item $i$ where $x_{ij} \geq 0$, ordered by non-increasing size $\ell(i, j)$, do
        $currtotal \leftarrow currtotal + x_{ij}$
        If $currtotal \leq 1$
            Add edge $(v_{js}, u_i)$ to $B$
            $x'(v_{js}, u_i) \leftarrow x_{ij}$
            If $currtotal = 1$ $s \leftarrow s + 1, currtotal \leftarrow 0$
        Else
            Add edge $(v_{js}, u_i)$ to $B$
            $x'(v_{js}, u_i) \leftarrow x_{ij} - (currtotal - 1), currtotal \leftarrow currtotal - 1, s \leftarrow s + 1$
            Add edge $(v_{js}, u_i)$ to $B$
            $x'(v_{js}, u_i) \leftarrow currtotal$
        End If
    End For
End For
```

This construction connects each node $v_{js} \in V$ to one or more nodes in U, so that for any bin j, the sum of x' vector over edges incident to nodes v_{js} for $s = 1, \ldots, k_j - 1$ is 1.

Definition 3.1 *A non-negative vector x on the edges of a graph is a* fractional matching *if, for each node u, the sum of components of x corresponding to edges incident on u is at most 1. The fractional matching* exactly matches a node u *if the corresponding sum is exactly 1. A fractional matching x is a* matching *if each component of x is 0 or 1.*

It is easy to verify that the graph B and the vector x' have the following properties:

- x' is a fractional matching in B of cost at most C.
- x' exactly matches each item node u_i for $i = 1, \dots, n$.
- x' exactly matches each bin node v_{js} for $j = 1, \dots, m$ and $s = 1, \dots, k_j - 1$.

The next step of the algorithm is finding a minimum-cost integral matching M in B that exactly matches all item nodes in U. This matching defines the required pseudopacking: assign item i to bin j for each edge $(v_{js}, u_i) \in M$.

Let us show that this pseudopacking satisfies the requirements of Theorem 3.1. Observe that since there exists a fractional matching in B of cost at most C (given by vector x'), and since the integrality gap for matching is 1, there exists an integral matching M with cost at most C. It remains to show that the total size of items packed into each bin satisfies the pseudopacking requirements. Define $\ell_{js}^{\max}$ to be the maximum item size $\ell(i,j)$ corresponding to edges incident on v_{js}, and similarly let $\ell_{js}^{\min}$ denote the minimum item size $\ell(i,j)$ corresponding to edges incident on v_{js}. Clearly, $\ell_{js}^{\min} \geq \ell_{j,s+1}^{\max}$ for all $j \in J, s = 1, \dots, k_j$. Denote the *adjusted capacity* for each bin j to be $c'(j) = \min(T, c(j))$. The total size of items packed into bin j is at most $\sum_{s=1}^{k_j} \ell_{js}^{\max}$. By the constraints of (1), x_{ij} may be positive only where $\ell_{ij} \leq c'(j)$, so $\ell_{j1}^{\max} \leq c'(j)$. The sum of the remaining terms for each bin j is:

$$\sum_{s=1}^{k_j} \ell_{js}^{\max} \leq \sum_{s=2}^{k_j-1} \ell_{js}^{\min} \leq \sum_{s=1}^{k_j-1} \sum_{i:(v_{js},u_i)\in E} \ell_{ij} x'(v_{js}, u_i) \leq$$

$$\sum_{s=1}^{k_j} \sum_{i:(v_{js},u_i)\in E} \ell_{ij} x'(v_{js}, u_i) = \sum_{i=1}^{n} \ell_{ij} x_{ij} \leq c'(j)$$

Therefore the total size of the items packed into bin j is at most $2c'(j)$. Furthermore, the removal of at most one item (the one matched to bin node v_{j1}) is sufficient to convert the pseudopacking into a feasible packing.

This completes the proof of Theorem 3.1.

The time complexity of the algorithm used in this proof is dominated by the solution to the linear program (1).

3.2 A (1,2) Bi-criteria Approximation for Min-GAP

Theorem 3.2 *There exists a (1,2) bi-criteria approximation for Min-GAP, i.e. a polynomial-time algorithm yielding a feasible assignment of all items to bins with cost equal to the smallest possible cost, and with makespan of at most twice the smallest makespan for solutions not exceeding that cost.*

Proof: The total cost of any feasible assignment is in the range $[0, nW]$, where $W = \max_{i,j} w(i,j)$. The makespan of any feasible assignment does not exceed $T_{max} = n \times \max_{i,j} \ell(i,j)$. Using a bisection search within the range of possible total costs, we find the smallest cost C for which there exists a feasible fractional solution to $LP(C, T_{max})$. This requires solving the linear program at most $\log(nW)$ times. Denote the cost of the fractional solution found by this search by C_{frac}.

Next, we use a bisection search on the range $[0, T_{max}]$ to find the smallest makespan T for which there exists a feasible fractional solution to $LP(C_{frac}, T)$. This requires solving the linear program at most $\log(T_{max})$ times. Denote the makespan of the fractional solution found by this search by T_{frac}.

We apply the algorithm of Theorem 3.1 to obtain a pseudopacked solution with total cost C_{frac} and makespan of at most $2T_{frac}$. The proof is completed by observing that C_{frac} can not be larger than the smallest possible cost of any integer assignment, and T_{frac} can not be larger than the smallest possible makespan of any integer assignment not exceeding a total cost of C_{frac}.

The algorithm described here solves a linear program at most $\log(nW) + \log(T_{max})$ times, followed by applying the algorithm of Theorem 3.1. Both of these steps may be performed in polynomial time.

□

3.3 Approximating GAP

Chekuri and Khanna [3] proved:

Theorem 3.3 ([3]) *There exists a 1/2-approximation algorithm for GAP.*

Proof: As previously mentioned, the term GAP in this work refers to the maximization problem as in Definition 2.1.

We show how to transform an instance of GAP to an instance of Min-GAP. Each profit on the GAP instance is converted into a corresponding cost in the Min-GAP instance by setting $w(i,j) = L - p(i,j)$, where $L > \max_{i,j} p(i,j)$ is selected so that all costs are positive. To create a feasible instance, we add a bin b_{m+1} with capacity 0, and for all items i we set the item sizes $\ell(i, m+1) = 0$ and the costs $w(i, m+1) = L$. We then use the algorithm from the proof of Theorem 3.2 to find a pseudo-packing. This pseudo-packing is transformed into an assignment - not necessarily feasible - for the GAP instance, as follows: For each item packed by the psuedo-packing into bin j, $1 \leq j \leq m$, assign the item to bin j in the GAP instance.

Theorem 3.2 guarantees that this assignment has at least the optimum profit. To obtain a feasible assignment that has at least half the profit, we examine each bin whose capacity is violated by the assignment. Let j be any such bin, and let i_j be the item whose removal assures feasibility as guaranteed by Theorem 3.1. If $p(i_j, j)$ is at least half the total profit of bin j we discard all other items assigned to j, otherwise we discard item i_j from the assignment. This gives a feasible assignment with at least half the profit given by the LP solution.

$\square$

4 A $(1-1/e)$ Approximation Algorithm for Fixed-Profit GAP

This section presents our main result:

Theorem 4.1 *Fixed-Profit GAP admits a $(1-1/e)$ approximation algorithm.*

Let J be the set of m bins, and I be the set of n items. Recall that each bin j has capacity $c(j)$, and each item i has a bin-dependent size $\ell(i,j)$ and a global profit $p(i)$. We use a standard LP-formulation for set-packing problems. For $S \subseteq I$ let $p(S) = \sum_{i \in S} p(i)$ and for a bin j, let $\ell(S,j) = \sum_{i \in S} \ell(i,j)$. Let $\Pi = \{(S,j) : \ell(S,j) \leq c(j)\}$. For every pair $(S,j) \in \Pi$ introduce a variable $y_{(S,j)}$ which represents the "amount of S packed in the bin j". Then integral feasible solutions to the following linear program correspond to feasible solutions to the Max-GAP with fixed profits instance.

$$\begin{array}{lll} \max & \sum_{(S,j)\in\Pi} p(S) y_{(S,j)} & \\ \text{s.t.} & \sum_{(S,j)\in\Pi, i\in S} y_{(S,j)} \leq 1 & \forall i \in I \\ & \sum_{(S,j)\in\Pi} y_{(S,j)} \leq 1 & \forall j \in J \\ & y_{(S,j)} \geq 0 & \forall (S,j) \in \Pi. \end{array} \tag{2}$$

The corresponding dual problem is:

$$\begin{array}{lll} \min & \sum_{i\in I} x_i + \sum_{j \in J} z_j & \\ \text{s.t.} & \sum_{i\in S} x_i + z_j \geq p(S) & \forall (S,j) \in \Pi \\ & x_i, z_j \geq 0 & \forall i \in I, j \in J \end{array} \tag{3}$$

Note that (3) has exponential number of constraints, while (2) has exponential number of variables. However, any basic feasible solution of (2) has at most $n+m$ non-zero variables. Now, if we had a polynomial time separation oracle for (3), we could compute an optimal solution to (2) (the non-zero entries) in polynomial time, see Chapter 6 in [7]. The number of non-zero entries in such a computed solution is polynomial in $n+m$. Unfortunately, such an oracle may not exist, since the separation problem for (3) defined by a specific bin j is equivalent to the knapsack problem. To see this, introduce new variables $w_i = p(i) - x_i$ and rewrite the constraints in (3) as $w(S) = \sum_{i \in S} w_i \leq z_j$. Then, checking whether for a specific j there exists $(S,j) \in \Pi$ so that $w(S) > z_j$ is equivalent to checking whether there exists $S \subseteq I$ so that $\ell(S,j) \leq c(j)$ and $w(S) > z_j$. The latter is a knapsack problem. Since knapsack admits an FPTAS, we get an *approximate separation oracle*, which for any $\varepsilon > 0$ checks whether there exists $(S,j) \in \Pi$ so that $w(S) > z_j(1-\varepsilon)$. This implies that we can solve the following linear program in time polynomial in $1/\epsilon$ and in its size:

$$\begin{array}{lll} \min & \sum_{i \in I} x_i + \sum_{j \in J} z_j & \\ \text{s.t.} & \sum_{i \in S} x_i + z_j(1-\varepsilon) \geq p(S) & \forall (S,j) \in \Pi \\ & x_i, z_j \geq 0 & \forall i \in I, j \in J. \end{array} \tag{4}$$

Thus we can also solve the dual of (4), which is:

$$\begin{array}{lll} \max & \sum_{(S,j) \in \Pi} p(S) y_{(S,j)} & \\ \text{s.t.} & \sum_{(S,j) \in \Pi, i \in S} y_{(S,j)} \leq 1 & \forall i \in I \\ & \sum_{(S,j) \in \Pi} y_{(S,j)} \leq \frac{1}{1-\varepsilon} & \forall j \in J \\ & y_{(S,j)} \geq 0 & \forall (S,j) \in \Pi. \end{array} \tag{5}$$

Let ν and $\nu(\varepsilon)$ denote the optimal values of (2) and of (5), respectively. Clearly, $\nu(\varepsilon) \geq \nu$. Note that if $y(\varepsilon)$ is a feasible solution to (5) then $(1-\varepsilon)y(\varepsilon)$ is a feasible solution to (2). In particular, a feasible solution $y = (1-\varepsilon)y(\varepsilon)$ to (2) of value at least $(1-\varepsilon)\nu(\varepsilon) \geq (1-\varepsilon)\nu$ can be found in time polynomial in $1/\varepsilon$ and in the size of the problem. We will apply a standard randomized rounding on y to obtain an integral feasible solution $\tilde{y}$ to (2). The rounding procedure is as follows.

1. For each bin j choose randomly with distribution $y_{(S,j)}$ a unique set A_j of items assigned to bin j at this stage (possibly $A_j = \emptyset$).

2. For every item assigned to more than one bin, remove that item from all bins containing it, except for one, chosen arbitrarily.

Let $\tilde{y}$ be an integral solution derived from y by such randomized rounding, and let $\tilde{\nu} = \sum_{(S,j) \in \Pi} p(S)\tilde{y}_{(S,j)}$ be the (random variable corresponding to the) profit of $\tilde{y}$. Let P_i be the probability that item i is packed by $\tilde{y}$, that is, P_i is the probability that there exists $(S,j) \in \Pi$ with $i \in S$ and $\tilde{y}_{(S,j)} = 1$. Let $x_{ij} = \sum_{(S,j) \in \Pi, i \in S} y_{(S,j)}$ be the fraction of item i assigned to bin j by y, and let $x_i = \sum_{j \in J} x_{ij}$ be the overall fraction of item i assigned to all bins by y.

Proposition 4.2 *$P_i \geq (1 - 1/e + 1/(32m^2))x_i$ for every item $i \in I$.*

Proof: Let $i \in I$. Clearly, $P_i = 1 - \Pi_{j \in J}(1 - x_{ij})$. Let $T = \{j \in J : x_{ij} > 0\}$ be the set of bins $j \in J$ with x_{ij} being non-zero, and let $t = |T|$. Notice that $t \leq m = |J|$, and we may clearly assume that $t \geq 2$ since for $t = 1$ the statement to prove is trivial. The minimum value of $1 - \Pi_{j \in J}(1 - x_{ij})$ is attained when $x_{ij} = x_i/t$ for every $j \in T$. Thus $P_i \geq 1 - (1 - x_i/t)^t$, and

$$\frac{P_i}{x_i} \geq \frac{1}{x_i}\left(1 - \left(1 - \frac{x_i}{t}\right)^t\right).$$

The right-hand side of the latter inequality is monotonically decreasing in x_i, so its minimum is reached when $x_i = 1$. Thus

$$\frac{P_i}{x_i} \geq \frac{1}{x_i}\left(1 - \left(1 - \frac{x_i}{t}\right)^t\right) \geq 1 - \left(1 - \frac{1}{t}\right)^t.$$

Since $t \leq m$, in order to complete the proof it suffices to show that for all $t \geq 2$, $(1 - 1/t)^t < 1/e - 1/(32t^2)$. To see this, let $\Delta(t) = 1/e - (1 - 1/t)^t$. Clearly $\Delta(t) > 0$ and $\Delta(t)$ is monotone decreasing. For $t \geq 2$

$$\begin{aligned}
\Delta(t) - \Delta(t+1) &= \left(1 - \frac{1}{t+1}\right)^{t+1} - \left(1 - \frac{1}{t}\right)^t \\
&= \left(1 - \frac{1}{t}\right)^t \left[\left(1 + \frac{1}{t^2 - 1}\right)^t \left(1 - \frac{1}{t+1}\right) - 1\right] \\
&> \left(1 - \frac{1}{t}\right)^t \left[\left(1 + \frac{t}{t^2 - 1}\right)\left(1 - \frac{1}{t+1}\right) - 1\right] \\
&= \left(1 - \frac{1}{t}\right)^t \left[\left(1 + \frac{t}{t^2 - 1} - \frac{1}{t+1} - \frac{t}{(t^2 - 1)(t+1)}\right) - 1\right] \\
&= (1 - \frac{1}{t})^t \frac{1}{(t^2 - 1)(t+1)} \\
&\geq \frac{1}{4(t^2 - 1)(t+1)}
\end{aligned}$$

In particular,

$$\Delta(t) \geq \sum_{s=t}^{\infty} \frac{1}{4(s^2 - 1)(s+1)} > \frac{1}{32t^2}.$$

□

Note that we used a solution y of value $\sum_{i \in I} p(i)x_i \geq (1-\varepsilon)\nu$. Also note that since our algorithm is polynomial in $1/\varepsilon$ and the size of the input we may choose $\epsilon = 1/(32m^2)$ and our algorithm remains polynomial in the size of the input. It follows therefore that

$$\begin{aligned} E(\tilde{\nu}) &= \sum_{i \in I} p(i)P_i \geq \left(1 - \frac{1}{e} + \frac{1}{32m^2}\right) \sum_{i \in I} p(i)x_i \\ &\geq \left(1 - \frac{1}{e} + \frac{1}{32m^2}\right)(1-\varepsilon)\nu \\ &> \left(1 - \frac{1}{e} + \frac{1}{32m^2} - \varepsilon\right)\nu = \left(1 - \frac{1}{e}\right)\nu. \end{aligned}$$

To complete the proof of Theorem 4.1, we show that the algorithm can be derandomized. We show how the method of conditional probabilities (see, e.g., [2]) can be used in our setting. As before, let y be a feasible solution to (2) of value at least $(1-\varepsilon)\nu(\varepsilon) \geq (1-\varepsilon)\nu$. Now, for each bin $j \in J$ let $S(j) = \{S : y_{(S,j)} > 0\}$. In case $\nu_j = \sum_{(S,j) \in \Pi} y_{(S,j)} < 1$ we also add the empty set to $S(j)$ and define $y_{(\emptyset,j)} = 1 - \nu_j$. Recall that each $S(j)$ has only a polynomial size. When we preformed our randomized rounding we have selected a unique $A_j \in S(j)$ (each with its corresponding probability $y_{(S,j)}$) and proved that $E(\tilde{\nu}) = \sum_{i \in I} p(i)P_i > (1-1/e)\nu$. We shall now *deterministically* decide which $S \in S(j)$ to chose as A_j, starting sequentially from bin 1, until the last bin is reached, and A_m is selected. For $S \in S(1)$, let $E(\tilde{\nu} \mid S)$ denote the conditional expectation of $\tilde{\nu}$ *given* that we selected S as A_1. By the formula for conditional probabilities we have

$$E(\tilde{\nu}) = \sum_{S \in S(1)} E(\tilde{\nu} \mid S) y_{(S,j)}.$$

In particular, there *exists* $S \in S(1)$ for which $E(\tilde{\nu} \mid S) \geq E(\tilde{\nu})$. How do we locate this S? We first compute $E(\tilde{\nu})$ *precisely.* This can be done since $E(\tilde{\nu}) = \sum_{i \in I} p(i)P_i$ and since each P_i can be computed precisely as each x_{ij} is known. We sequentially test all $S \in S(1)$ (there are only a polynomial number of tests). For each $S \in S(1)$ we can compute *precisely* $E(\tilde{\nu} \mid S)$ because

$$E(\tilde{\nu} \mid S) = p(S) + \sum_{i \in I \setminus S} p(i)(1 - \Pi_{j \in J \setminus \{1\}}(1 - x_{ij})).$$

Thus, we can deterministically locate $S \in S(1)$ for which $E(\tilde{\nu} \mid S) \geq E(\tilde{\nu})$ and define $A_1 = S$. We now continue to bins $2, 3, \ldots, m$ and do the same thing while maintaining the invariant that the conditional expectation, given the selections in the prior bins, never deceases. After selecting the set A_m in the last bin we have a deterministic selection whose profit is at least as good as the original expectation $E(\tilde{\nu})$, and hence at least $(1-1/e)\nu$. This completes the proof of Theorem 4.1. The proof also shows that the integrality gap of (2) is at least $(1-1/e)$, that is, there always exists an integral solution to (2) of value at least $(1-1/e)\nu$.

We remark that our algorithm extends to a more general problem, MKARCC, described in section 2, as shown in the following corollary.

Corollary 4.3 *Max-GAP with Color Constraints Problem with fixed profits admits a* $(1-1/e)$*-approximation algorithm.*

Proof: The proof uses the same strategy as the one of Theorem 4.1. The definition of Π needs to be changed so that it includes only those sets of items that do not violate the color constraints. The approximate separation oracle for a specific constraint related to a specific bin checks all possible combinations of colors for the bin, and finds the approximate solution for the knapsack problem when the set of items under consideration is restricted to each color combination. If some color combination violates the constraint, then we have found a pair (S, j) violating the constraint. The number of combinations is at most $|I|^K$. Therefore, the approximate separation oracle is still performed in polynomial time. No other part of the proof is affected. $\square$

5 The MKAR Problem

5.1 Special Instances of MKAR are APX-hard

We show that even highly restricted instances of MKAR are APX-hard. We note that the case when $\ell(i) \in \{1, 2\}$ (but bin capacities and item profits are arbitrary) was shown to be APX-hard in [1].

Theorem 5.1 *MKAR with unit profits is APX-hard even on instances where all the bins have size* 3 *and* $\ell(i) \in \{1, 3\}$ *for all* i.

Proof: The proof is similar to the one given in [3] to show APX-hardness of some Max-GAP instances, and is presented here to show that MKAR, a more restricted problem than Max-GAP, is also APX-hard. The following problem is reduced to MKAR:

Maximum 3-Dimensional Matching (3DM):
Instance: An equitable partition X, Y, Z of a ground set V and a set-family $T \subseteq X \times Y \times Z$.
Objective: Find a subfamily $M \subseteq T$ of pairwise disjoint sets (matching) of maximal size.

Here is the reduction. Given an instance $((X, Y, Z), T)$ of 3DM with $|T| = m$ and $|X| = |Y| = |Z| = n$, create an instance of MKAR as follows. The set of bins is T, and the set of items is $V \cup U$ where U is a set of additional $m - n$ items. All bins have size 3, items in V have size 1, items in U have size 3, and all items have unit profits. Item $i \in V$ can be placed in bin $j \in T$ if the set j contains the element i. Items in U can be placed in any bin. Clearly, 3 items from V can fit in a bin if, and only if, they form a set in T. Thus bins with 3 items correspond to a matching in T. It follows therefore that if 3DM has a matching of size n, then MKAR has a solution of size $3n + (m - n) = 2n + m$.

A 3-bounded instance 3DM-3 of 3DM is one in which the number of occurrences of any element of V in T is at most 3. Kann [9] showed that there exists an $\varepsilon_0 > 0$ such that it is NP-hard to decide whether an instance of 3DM-3 has a matching of size n or if every matching has size at most $(1 - \varepsilon_0)n$. In the latter case, our MKAR instance can achieve a profit of at most $3(1 - \varepsilon_0)n + 3\varepsilon_0 n + [m - (1 - \varepsilon_0)n - 3\varepsilon_0 n/2] = 2n + m - \varepsilon_0 n/2$.

It follows therefore that it is NP-hard to decide whether our MKAR instance can achieve a profit of $2n + m$ or of $2n + m - \varepsilon_0 n/2$. The APX-hardness now follows from the fact that $m = O(n)$ for 3DM-3. □

5.2 A Combinatorial $1/2$-Approximation Algorithm for MKAR

We give a simple and fast combinatorial 1/2-approximation algorithm for MKAR which does not require solving linear programs (as in Theorem 4.1 or as in [11]), generalizing [5].

Theorem 5.2 *MKAR has a $1/2$-approximation algorithm whose running time $O(T(m,n)+m^2n^2)$, where $m=|J|$, $n=|I|$, and $T(m,n)$ is the time for computing a maximum cost (s,t)-flow in a capacitated bipartite graph with parts of size m and n.*

Given an instance of MKAR, the *assignment graph* is a bipartite graph $G=(I+J,E)$ where $ij \in E$ if $i \in I(j)$ (assuming $\ell(i) \le c(j)$ for every $i \in I(j)$). Consequently, $I(j)$ is the set of neighbors of j in G. Each edge $e=ij \in E$ has length $\ell_e=\ell(i)$ and profit $p_e=p(i)$. Let $\pi_e=p_e/\ell_e$. For a node v of G let $\delta(v)$ denote the set of edges incident to v. Consider the linear program:

$$\begin{array}{lll} \max & \sum_{e\in E}\pi_e x_e & \\ \text{s.t.} & \sum_{e\in\delta(j)} x_e \le c(j) & \forall j \in B \\ & \sum_{e\in\delta(i)} x_e \le \ell(i) & \forall i \in I \\ & x_e \ge 0 & \forall e \in E. \end{array} \tag{6}$$

Let x be a feasible solution to (6). We say that x_e (or e) is *fractional* if $0<x_e<\ell_e$ (for $e=ij$, x_e is the amount of item i packed in bin j). Any non-fractional feasible solution to (6) bijectively corresponds to a solution to the MKAR instance, and has profit $\sum_{e\in E}\pi_e x_e$.

Lemma 5.3 *Let x be a basic feasible solution to (6). Then the set $F(x)$ of fractional edges is a forest so that in any connected component of $F(x)$ at most one leaf belongs to I. In particular, if M is a maximum matching in $F(x)$, then any non-isolated node $i \in I$ of $F(x)$ is matched by M.*

Proof: Let x be a feasible solution to (6), and let $P=(e_1,\ldots,e_\ell)$ be a path or a cycle in $F(x)$. Let $P'=\{e_1,e_3,\ldots\}$, $P''=P-P'=\{e_2,e_4,\ldots\}$. Set $\varepsilon=\min\{\varepsilon^+,\varepsilon^-\}$ where $\varepsilon^+=\min\{\ell_e-x_e : e\in P\}$ and $\varepsilon^-=\min\{x_e : e \in P\}$. Since all the edges in P are fractional, $\varepsilon>0$. Let $d_e=\varepsilon$ for $e\in P'$, $d_e=-\varepsilon$ for $e \in P''$, and $d_e=0$ otherwise. Let $x'=x+d$, $x''=x-d$. It is easy to see that if P is a cycle, then x',x'' are feasible solutions. Since $x=(x'+x'')/2$, x cannot be basic. Thus $F(x)$ is a forest. A similar argument applies if P is a path in $F(x)$ between two leaves $u,v\in I$. In this case, it is easy to see that x',x'' satisfy the constraints of (6) for every node of G distinct from u,v. To see that this is so for u and for v, note that if $i\in I$ is a leaf of $F(x)$, then for $e\in\delta(i)$, $x_e>0$ if, and only if, e is the unique edge of $F(x)$ incident to i. Consequently, in any connected component of $F(x)$ at most one leaf belongs to I. The second statement follows easily from the first statement. $\square$

Corollary 5.4 *Given a feasible solution x to (6) we can find in $O(|E(G)|^2)$ time a feasible solution z such that $\pi \cdot z \geq \pi \cdot x$ and $F(z)$ is a forest so that in any connected component of $F(z)$ at most one leaf belongs to I, where $|E(G)|$ is the number of edges in G.*

Proof: For $Q \subseteq E$ denote $\pi(Q) = \sum\{\pi(e) : e \in Q\}$. Let P, P', P'' be as in the proof of Lemma 5.3. Set $Q = P'$ if $\pi(P') \geq \pi(P'')$ and $Q = P''$ otherwise, and $\varepsilon = \min\{\varepsilon^+, \varepsilon^-\}$, where $\varepsilon^+ = \min\{\ell_e - x_e : e \in Q\}$ and $\varepsilon^- = \min\{x_e : e \in P - Q\}$. Let $z_e = x_e + \varepsilon$ for $e \in Q$, $z_e = x_e - \varepsilon$ for $e \in P - Q$, and $z_e = x_e$ otherwise. Note that $|F(z)| \leq |F(x)| - 1$ (by the choice of ε), and that $\pi \cdot z \geq \pi \cdot x$. Furthermore, z is a feasible solution to (6) if x is, by an argument similar to the one as in the proof of Lemma 5.3. Thus, by repeatedly replacing x by z as above, we convert x into z as in the statement. Finding cycle/path P as above (or determining that such does not exist) can be done in $O(|E(G)|$ time, and each time P is found the number of fractional edges reduces by at least 1. The statement follows. $\square$

Lemma 5.5 *Given an instance of MKAR, a pseudopacking of profit at least the optimal value of (6) can be computed in $O(T(m,n) + m^2n^2)$ time, where m, n and $T(m,n)$ are as in Theorem 5.2.*

Proof: Finding an optimal solution x to (6) (which may not be basic) is easily reduced to a max-cost flow problem as follows. We direct all edges in G from I to J, add a source s with edges si of the capacity $\ell(i)$ for every $i \in I$, and a sink t with edges jt of the capacity $c(j)$ for every $j \in J$. The costs are π_e for $e \in E$ and are zero otherwise. We then compute a flow f of maximum cost from s to t. It is easy to see that the restriction of f to E is an optimal solution to (6). Then we compute in $O(|E(G)|^2) = O(m^2n^2)$ time a feasible solution z as in Corollary 5.4. Finally, we compute a maximum matching M in $F(z)$ and set $x_e = \ell_e$ for every $e \in M$. It is easy to see that the resulting (possibly non-feasible) solution that has no fractional edges corresponds to a pseudopacking as required. $\square$

As was shown in section 3.3, after a pseudopacking of profit at least opt is found, an assignment of profit at least opt/2 can be found in linear time. Thus Lemma 5.5 implies Theorem 5.2.

We note that the integrality gap of (6) is 1/2 even for unit profits. Note that in the previous section we used a different LP-formulation to get a better approximation ratio.

An example of the execution of this algorithm is given in Figure 1. Figure 2 gives a case for which the 1/2 approximation ratio is tight.

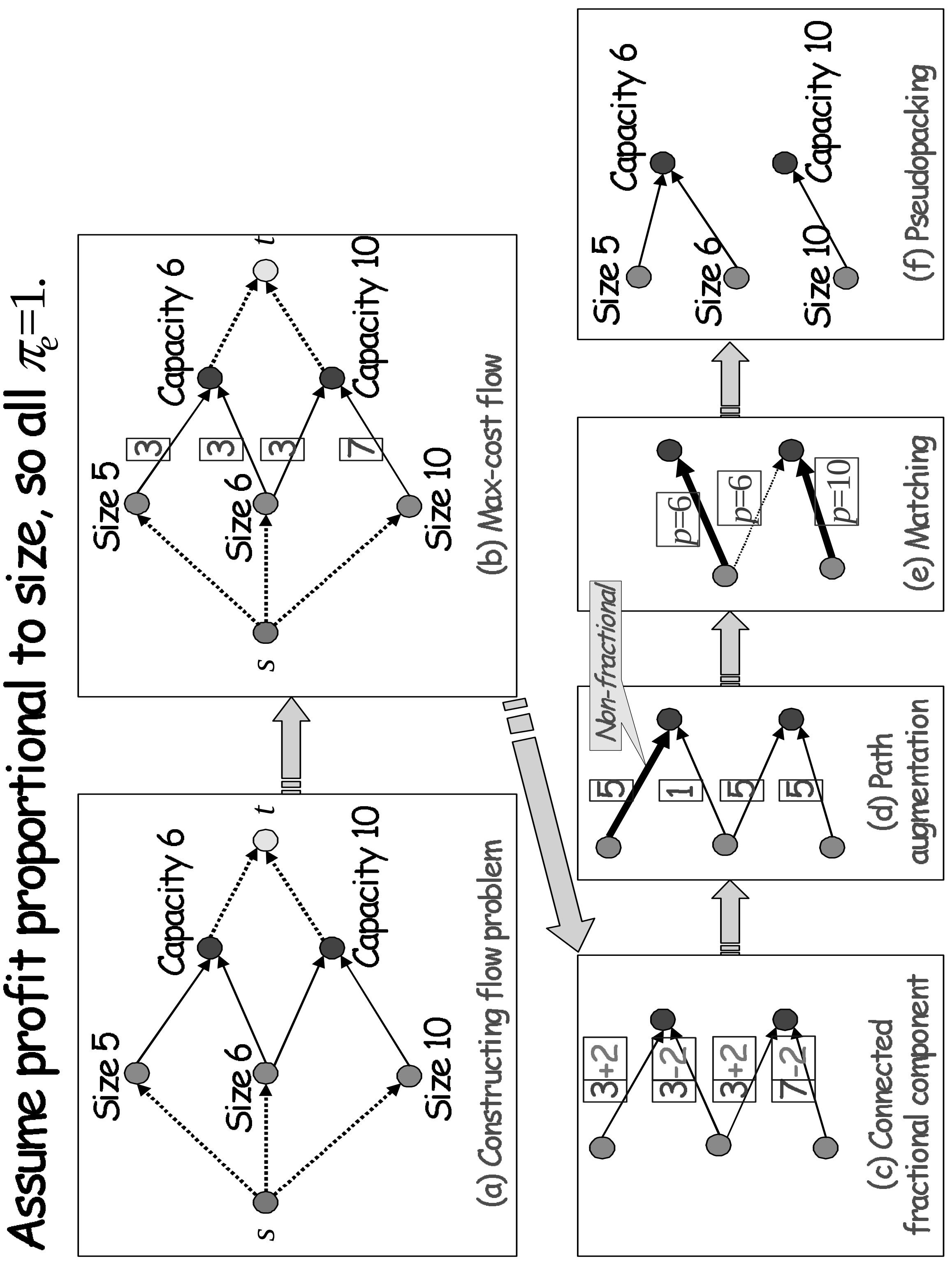

Figure 1: Example for the algorithm used in proving Theorem 5.2

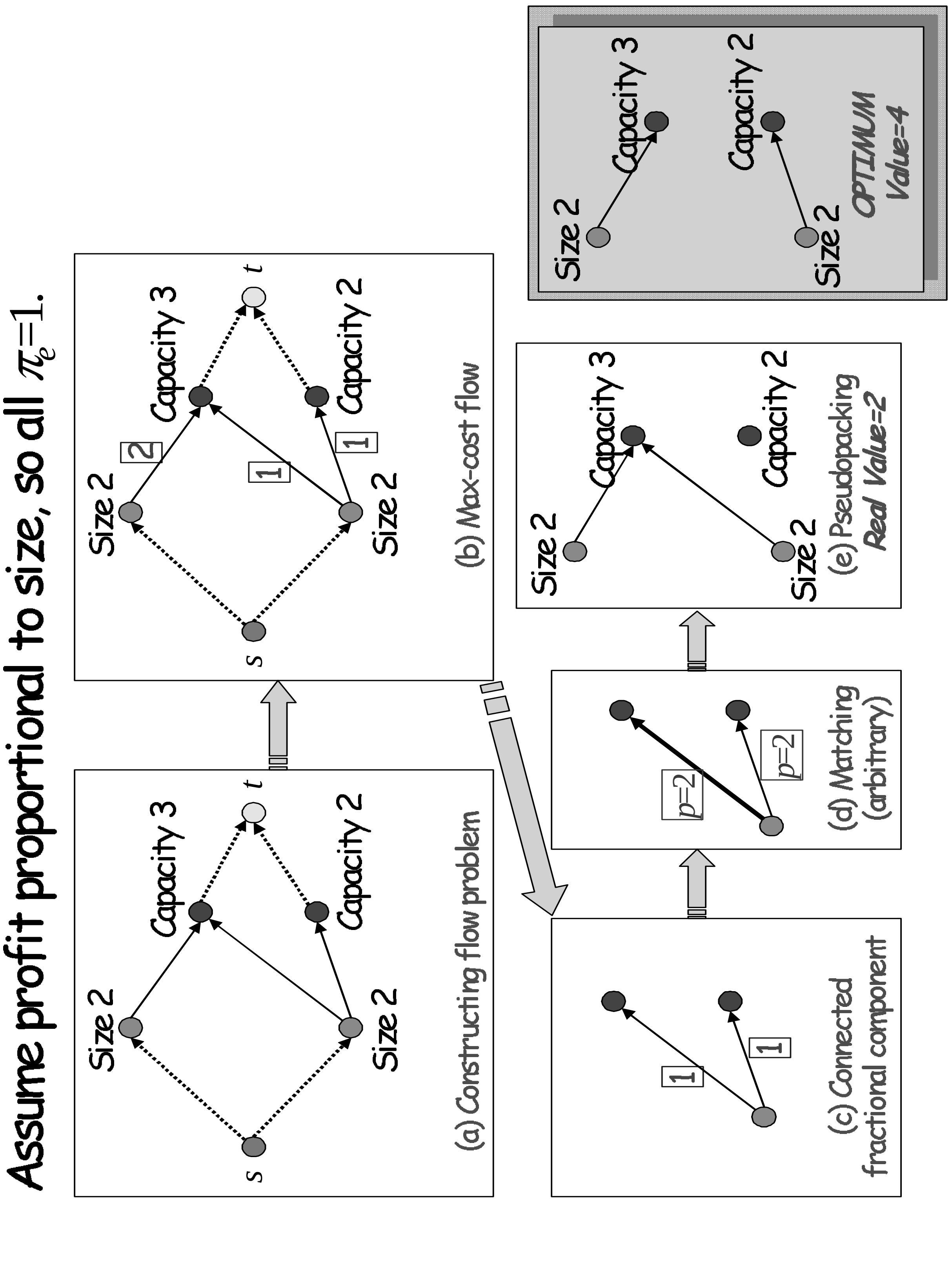

Figure 2: Tight example for the algorithm used in proving Theorem 5.2

6 Conclusions

In this work, we have shown the following results: Our main result is a $(1-1/e)$-approximation for Fixed-Profit GAP. We also gave a fast combinatorial $(1/2)$-approximation for MKAR, which is a restriction of Fixed-Profit GAP.

One open problem is extending our $(1-1/e)$-approximation to GAP, without restrictions on profits, or proving that such an algorithm is unlikely to exist.

Another open problem is determining whether a better approximation ratio for Fixed-Profit GAP is possible. There is a well-known problem - the Max-Coverage problem - for which $(1-1/e)$ was shown to be the best possible approximation ratio, unless P=NP [6]. We conjecture that this may be also shown for Fixed-Profit GAP, possibly by using similar methods as in [6].

References

[1] J. Aerts, J. Korst, F. Spieksma, W. Verhaegh, and G. Woeginger, Random redundant storage in disk arrays: complexity of retrieval problems, IEEE Transactions on Computers, Vol. 52, no. 9 (2003), 1210–1214.

[2] N. Alon and J.H. Spencer, *The Probabilistic Method, Second Edition*, Wiley, New York, 2000.

[3] C. Chekuri and S. Khanna, A PTAS for the multiple knapsack problem, SODA 2000, 213–222.

[4] M. Dawande and J. Kalagnanam, The multiple knapsack problem with color constraints, *IBM Research Report*, RC 21128 (94508), Revised March 24, 1998.

[5] M. Dawande, J. Kalagnanam, P. Keskinocak, R. Ravi, and F. S. Salman, Approximation algorithms for the multiple knapsack problem with assignment restrictions, *J. of Combinatorial Optimization*, 2000, Vol. 4,171–186.

[6] U. Feige, A threshold of $\ln n$ for approximating set cover, *J. ACM*, Vol. 45,634–652.

[7] M. Grötchel, L. Lovasz, and A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer, 1998.

[8] M. S. Hung and J. C. Fisk, A heuristic routine for solving large loading problems, *Naval Research Logisitical Quarterly*, 26(4):643-650, 1979.

[9] V. Kann, Maximum Bounded 3-dimensional Matching is MAX SNP-Complete, *Information Processing Letters* 37, 1991, 27–35.

[10] H. Shachnai, T. Tamir, Polynominal time approximation schemes for class-constrained packing problem, *APPROX 2000*, 238–249.

[11] D. B. Shmoys and E. Tardos, An approximation algorithm for the generalized assignment problem, *Math Programming A*, 62:461–74, 1993.

[12] J. Turek, J. Wolf and P. Yu, Approximate algorithms for scheduling parallelizable tasks, *SPAA '92*, 323–332.

[13] J. Wolf, P. Yu and H. Shachnai, Disk load balancing for video-on-demand systems, *ACM Multimedia Systems Journal*, 5:358–370, 1997.