Operations Research Letters 40 (2012) 282–286

Contents lists available at SciVerse ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

# Approximating the multi-level bottleneck assignment problem[☆]

Trivikram Dokka [a,*], Anastasia Kouvela [b], Frits C.R. Spieksma [a]

[a] ORSTAT, K.U.Leuven, Naamsestraat 69, B-3000, Leuven, Belgium

[b] Management Science Group, London School of Economics, UK

ARTICLE INFO

*Article history:*
Received 19 December 2011
Accepted 8 April 2012
Available online 21 April 2012

*Keywords:*
Bottleneck problem
Multidimensional assignment
Approximation
Computational complexity
Efficient algorithm

ABSTRACT

We consider the multi-level bottleneck assignment problem (MBA). This problem is described in the recent book "Assignment Problems" by Burkard et al. (2009) on pages 188–189, where its complexity status is called open. We view the problem as a special case of a bottleneck $m$-dimensional multi-index assignment problem and settle its complexity status. We give approximation algorithms and inapproximability results, depending upon the completeness of the underlying graph.

## 1. Introduction

We consider an $m$-dimensional bottleneck assignment problem. Let $V_1, V_2, \ldots, V_m$ be pairwise disjoint sets, each with cardinality $n$, and let $V = \cup_{i=1}^{m} V_i$. There is a given weight $w(v) \in \mathbb{N}$ for each $v \in V$. The set $V$ is the node-set of an $m$-partite graph that has a given set of arcs $E$ of the following form: $E = \{(u, v)|u \in V_i, v \in V_{i+1}, i = 1, \ldots, m-1\}$. Thus, an arc in $E$ connects a node from $V_i$ with a node from $V_{i+1} (1 \le i \le m-1)$, and there are no other arcs. A feasible $m$-tuple (to which we will refer as a *duty*) is a set of nodes $D = \{v_1, v_2, \ldots, v_m\}$ such that $v_i \in V_i$ for $1 \le i \le m$, and such that $(v_i, v_{i+1}) \in E$. The cost of a duty $D$ equals $c(D) = \sum_{v \in D} w(v)$. The problem is to find a partition of $V$ into $n$ duties $D_1, D_2, \ldots, D_n$ such that $\max_j c(D_j)$ is minimum. We will refer to this partition of $V$ into $\{D_1, D_2, \ldots, D_n\}$ as a solution $S$, and the cost of a solution $S$ equals $c(S) = \max_j\{c(D_j)|S = \{D_1, D_2, \ldots, D_n\}\}$.

This problem is known as the multi-level bottleneck assignment problem (MBA); obviously, for $m = 2$ a (special case of the) bottleneck assignment problem arises. It was introduced by Carraresi and Gallo [5], motivated by an application in bus driver scheduling. In the context of this application, a set $V_i$ corresponds to the shifts that need to be carried on day $i$ $(1 \le i \le m)$; further, an edge $(v_i, v_{i+1}) \in E$ indicates that it is possible to perform shift $v_{i+1}$ directly after shift $v_i$ $(1 \le i \le m-1)$. Then, a duty $D_j$ is a set of shifts, one from each day, to be carried out by a driver. The cost of a duty is nothing else but the load of a driver, and the goal is to minimize the maximum load. Notice that [5] phrase the problem using a weight $w(v)$ on each arc leaving $v$, when $v \in \cup_{i=1}^{m-2} V_i$, and an arc with weight $w(u) + w(v)$ for each arc $(u, v) \in V_{m-1} \times V_m$. They show that the problem is NP-hard when $m$ is part of the input by a reduction from Even–Odd Partition, and they leave as an open problem the complexity for a fixed $m$. This MBA problem is also described in the recent book by Burkard et al. [4], in which it is stated that the complexity of this problem is unresolved for each fixed $m \ge 3$ (pp. 188–189); we will settle this question in Section 2.

*Related work*

As described above, the problem seems to be first introduced in [5], who, in addition to proving NP-completeness for arbitrary $m$, also describe a heuristic with computational experiments. Another heuristic, with better computational results, is given in Bianco et al. [2], see also Aringhieri and Cordone [1].

An important special case of MBA is the case where $E$ is complete, i.e., in terms of the application, the case where each shift from day $V_i$ can be directly followed by each shift from day $V_{i+1}$. We will call this special case *complete-MBA*. In fact, this special case was studied by Hsu [9] and by Coffman and Yannakakis [6] from an approximation point of view. For complete-MBA, Hsu [9] gave an $\left(2 - \frac{1}{n}\right)$-approximation algorithm that runs in $O(mn \log n)$, while Coffman and Yannakakis [6] gave an $O(n^2 m)$ $\left(\frac{3}{2} - \frac{1}{2n}\right)$-approximation algorithm. For the case where $m = 3$, Hsu gave a $\frac{3}{2}$-approximation algorithm that runs in $O(n \log n)$, and a $\frac{4}{3}$-approximation algorithm that runs in $O(n^3 \log n)$. We will refer to our problem MBA with $m = 3$ as MBA3. Three-dimensional

[☆] An earlier version of this paper appeared in the proceedings of WALCOM'2012, LNCS 7157, 64–76.

[*] Corresponding author.
E-mail addresses: trivikram.dokka@econ.kuleuven.be (T. Dokka), A.Kouvela@lse.ac.uk (A. Kouvela), frits.spieksma@econ.kuleuven.be (F.C.R. Spieksma).

0167-6377/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.orl.2012.04.003

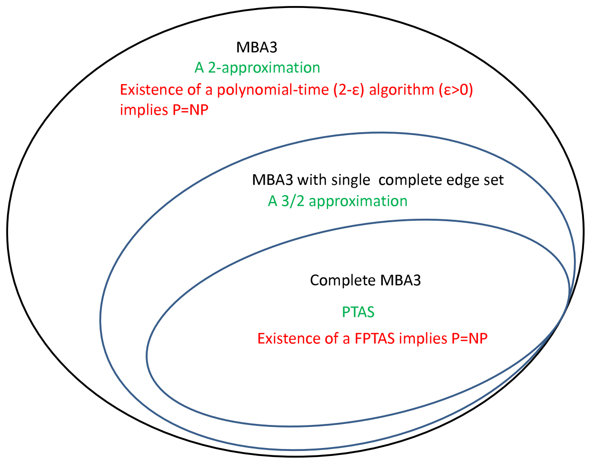
MBA3
A 2-approximation
Existence of a polynomial-time (2-ε) algorithm (ε>0) implies P=NP
MBA3 with single complete edge set
A 3/2 approximation
Complete MBA3
PTAS
Existence of a FPTAS implies P=NP

**Fig. 1.** Overview of results for *MBA3*.

bottleneck assignment problems with more general cost structures than MBA3 have been studied by Klinz and Woeginger [10], and more recently by Goossens et al. [8].

We observe here that MBA can also be seen as a generalization of the multi-processor scheduling problem, where one allows for incompatibilities between jobs. Indeed, when viewing each node $v \in V$ as a job with $w(v)$ its processing time, and each duty as a machine, then a multi-processor scheduling problem arises with the constraint that nodes (jobs) from the same set $V_i$ should not be assigned to the same duty (machine). Such problems have been studied in Bodlaender et al. [3]; however, their results do not apply to MBA.

As far as we are aware, all known approximation results deal with the case where every pair of nodes of different sets can be joined in a duty, i.e., the case of complete-MBA. In this paper we deal with a more general setting, namely the case where the edge set between $V_i$ and $V_{i+1}$ can be arbitrary (and not necessarily complete) ($1 \le i \le m-1$). Interestingly, it turns out that the approximation algorithms from [9,6] do not seem to generalize to this setting. In fact, we see as our main contribution exploring the approximability of MBA depending upon whether or not the edge sets between $V_i$ and $V_{i+1}$ are assumed to be complete. When we restrict ourselves to $m = 3$, notice that there is an edge set between $V_1$ and $V_2$, denoted by $E_1$, and an edge set between $V_2$ and $V_3$, denoted by $E_2$. Thus, when $m = 3$, we can distinguish three cases:

- both $E_1$ and $E_2$ are complete (complete-MBA3),
- $E_1$ or $E_2$ (or both) is complete (MBA3 with a single complete edge set), and
- both $E_1$ and $E_2$ are arbitrary (MBA3).

Observe that an instance of MBA3 is such that any node from $V_1$ can be joined in a duty with any node from $V_3$.

*Our results*

- There is a simple 2-approximation algorithm for MBA3 (see Section 3.1), while the existence of a $(2-\epsilon)$-polynomial time approximation algorithm for MBA3 ($\epsilon > 0$) implies $P = NP$ (see Section 3.2),
- There is $\frac{3}{2}$-approximation algorithm for MBA3 with a single complete edge set (see Section 4),
- There is a PTAS for complete-MBA for each $m \ge 3$ (see Section 5), while complete-MBA3 is strongly NP-hard (see Section 2),
- Online MBA can be seen as a generalization of online parallel machine scheduling (see Section 6).

See Fig. 1 for an overview of our results for MBA3. Notice that for complete-MBA3, and for MBA3, the proposed algorithms are best-possible.

## 2. The complexity of MBA

As already observed by [5], deciding feasibility of an instance of MBA is not a difficult problem: by verifying whether the arc set $E$ contains a bipartite matching between each pair of sets $(V_i, V_{i+1})$, $1 \le i \le m-1$, it follows whether a feasible solution exists. Finding a best solution, however, is a more difficult problem, even for $m = 3$, and even when the edge sets are complete. We will now informally argue that complete-MBA3 is at least as hard as Numerical 3-Dimensional Matching.

We first describe Numerical 3-Dimensional Matching (N3DM), which is known to be NP-hard. Numerical 3-Dimensional Matching has as input 3 sets of positive integers $x_1, \ldots, x_n$ (the set $X$), $y_1, \ldots, y_n$ (the set $Y$), and $z_1, \ldots, z_n$ (the set $Z$), and a bound $B$ such that $\sum_{i=1}^{n}(x_i + y_i + z_i) = nB$. The question is whether there exist $n$ disjoint triples, each containing one element from each of the three sets, such that for each triple $x_i + y_j + z_k = B$.

Given an instance of N3DM we now build, in a straightforward manner, an instance of our problem. Let the number of shifts equal $n$, and the number of days equal 3, i.e., $m := 3$. We assume that all arcs are present, i.e., $E := \{(u, v) | (u, v) \in (V_1 \times V_2) \cup (V_2 \times V_3)\}$. Further, the weight of a node $v_i \in V_1$ (or, in terms of [5], the weight of an arc leaving a node in $V_1$), equals $x_i$, thus $w(v_i) = x_i$ for each $v_i \in V_1$. Similarly, we have $w(v_i) := y_i$ for each $v_i \in V_2$, and $w(v_i) := z_i$ for each $v_i \in V_3$ (or, when phrased in terms of [5], the weight of an arc leaving node $v_i \in V_2$, going to node $v_{i'} \in V_3$ equals $y_i + z_{i'}$ $(1 \le i, i' \le n)$). The question is: does there exist a solution consisting of $n$ duties such that the maximum load of a duty is no more than $B$? This completes the description of an instance of MBA. It is now easy to verify that a yes-instance of N3DM corresponds to a solution of our problem with cost $B$, and vice versa. We record the above discussion by stating the following fact (which settles the question in [4]).

**Fact 1.** *Complete-*MBA3 *is NP-hard.*

## 3. The approximability of MBA3

In this section we focus on MBA3. We present a simple 2-approximation algorithm called Sequential Bottleneck (SB) in Section 3.1, and we show in Section 3.2 that this approximation factor is, in fact, best possible.

### 3.1. *The sequential bottleneck heuristic (SB)*

SB runs in two stages: In the first stage, SB computes a bottleneck matching between $V_1$ and $V_2$. More precisely, the following integer program is solved.

$$\begin{aligned}
\min \quad & \max_{u \in V_1, v \in V_2} (w(u) + w(v)) x_{u,v} \\
\text{s.t.} \quad & \sum_{v:\{u,v\} \in E} x_{u,v} = 1 \quad \text{for each } u \in V_1 \\
& \sum_{u:\{u,v\} \in E} x_{u,v} = 1 \quad \text{for each } v \in V_2 \\
& x_{u,v} \in \{0, 1\} \quad \text{for each } \{u, v\} \in E.
\end{aligned}$$

Let us denote the resulting matching as $M$, i.e., $M = \{(u, v) | x^*_{u,v} = 1\}$, and let $w(M)$ denote the weight of this matching. In the second stage, SB computes a bottleneck assignment between the pairs in $M$ and the elements in $V_3$, whose cost is equal to the cost of the resulting triple. Thus, we solve an integer program that is similar to the one above, where we replace $u \in V_1$ by $(u, v) \in M$, and where we replace $v \in V_2$ by $z \in V_3$. When heuristic SB is run on some instance $I$, we denote the cost of the resulting solution by SB($I$).

**Theorem 2.** *SB is a polynomial-time,* 2*-approximation algorithm for* MBA3*. Moreover, there exist instances for which this bound is tight.*

**Proof.** Obviously, SB is a polynomial-time algorithm, since it amounts to solving two bottleneck assignment problems (we refer to [4] for achievable time-bounds; when $E_1$ and $E_2$ are complete, simply sorting the weights suffices to solve the bottleneck assignment problems).

Consider now the solution found by SB; let its cost be determined by triple $(u, v, z) \in V_1 \times V_2 \times V_3$. Then:

$$SB(I) = w(u) + w(v) + w(z) \le w(M) + \max_{z \in V_3} w(z).$$

Further, it is easily seen that OPT $\ge w(M)$, and that OPT $\ge \max_{z\in V_3} w(z)$, where *OPT* refers to the cost of an optimal solution. The result follows.

Finally, consider the following example, where we — with a slight abuse of notation — identify elements of a set with their weights: $V_1 = \{n-1, \dots, 1, 0\}$, $V_2 = \{n-1, \dots, 1, 0\}$ and $V_3 = \{n, 0, \dots, 0\}$. The edge sets $E_1$ and $E_2$ are complete. The value of an optimal solution is $n$, while SB gives a solution with value $2n-1$. □

Notice that SB detects whether an instance has a feasible solution. Also, notice that in order to obtain a ratio of 2, any assignment in the second stage suffices. And although even more elaborate algorithms than SB can certainly be conceived, no polynomial time algorithm can improve upon the factor of 2 (unless $P = NP$), as we show next.

### 3.2. An inapproximability result

We show that MBA3 cannot be approximated within a factor of 2 unless $P = NP$. To do so, we use a traditional technique: we will show that a YES-instance of 3-dimensional matching (3DM) corresponds to an instance of MBA3 with cost 1, whereas a NO-instance corresponds to an instance of our problem with cost 2(or one that has no feasible solution). Then, a polynomial time approximation algorithm with a worst case ratio strictly less than 2 would be able to distinguish the YES-instances of 3DM from the NO-instances, and this would imply $P = NP$.

Let us first recall 3-dimensional matching.

Instance: Three sets $X = \{x_1, \dots, x_q\}$, $Y = \{y_1, \dots, y_q\}$, and $Z = \{z_1, \dots, z_q\}$, and a subset $T \subseteq X \times Y \times Z$.

**Question.** Does there exist a subset $T'$ of $T$ such that each element of $X \cup Y \cup Z$ is in exactly one triple of $T'$?

Let the number of triples be denoted by $|T| = p$. Further, let the number of triples in which element $y_j$ occurs, be denoted by $\#occ(y_j)$, $j = 1, \dots, q$.

Starting from arbitrary instance of 3DM, we now build a corresponding instance of MBA3 by specifying $V_i (i = 1, 2, 3)$, $E$, and the weights $w$ as follows:

- for each triple in $T$, there is a node in $V_2$. We refer to these nodes as *triple* nodes.
- for each $x_i \in X$, there is a node in $V_1$ $(i = 1, \dots, q)$. In addition, for each $y_j \in Y$, there are $\#occ(y_j) - 1$ nodes in $V_1$ $(j = 1, \dots, q)$; for such a node in $V_1$ we say that this node *corresponds* to element $y_j$. These latter nodes will be referred to as the *dummy* nodes of $V_1$.
- for each $z_k \in Z$, there is a node in $V_3$ $(k = 1, \dots, q)$. Further, we have $p - q$ additional nodes in $V_3$. These latter nodes will be referred to as the *dummy* nodes of $V_3$.

Notice that this construction ensures that $|V_2| = |V_1| = |V_3| = p$.

Let the nodes of $V_1, V_2$, and $V_3$ be denoted by $\{x'_1, \dots, x'_p\}$, $\{t'_1, \dots, t'_p\}$, and $\{z'_1, \dots, z'_p\}$ respectively. Thus, $\{x'_1, \dots, x'_q\}$ are the non-dummy nodes of $V_1$, and $\{x'_{q+1}, \dots, x'_p\}$ are the dummy nodes of $V_1$; notice that each dummy node of $V_1$ corresponds to some element $y_j \in Y$. Further, triple nodes $\{t'_1, \dots, t'_p\}$ simply correspond to the triples in $T$, while $\{z'_1, \dots, z'_q\}$ are the non-dummy nodes of $V_3$, and $\{z'_{q+1}, \dots, z'_p\}$ are the dummy nodes of $V_3$. The edge set $E$ is defined as follows:

- There is an edge $(x'_i, t'_j)$ if $x_i$ is in the $j$-th triple in $T$, for $i = 1, \dots, q$ and $j = 1, \dots, p$.
- There is an edge $(t'_j, z'_k)$ if $z_k$ is in the $j$-th triple in $T$, for $k = 1, \dots, q$ and $j = 1, \dots, p$.
- There is an edge $(t'_j, z'_k)$, for $j = 1, \dots, p$ and $k = q + 1, \dots, p$.
- There is an edge $(x'_i, t'_k)$ if element $y_j \in Y$, to which dummy node $x'_i$ corresponds, is contained in the $k$-th triple of $T$.

To complete the description of our instance of MBA3, we assign the weights to the nodes in $V_1, V_2$, and $V_3$ as follows. All weights are zero, except the weights of the dummy nodes in $V_1$, and the weights of the non-dummy nodes in $V_3$: these weights equal 1.

**Lemma 3.** *If the instance of* 3DM *is a YES-instance, the corresponding instance of* MBA3 *has cost* 1*. If the instance of* 3DM *is a NO-instance, the corresponding instance of* MBA3 *has cost* 2*, or is infeasible.*

**Proof.** Suppose that the instance of 3DM is a YES-instance. Then we construct a solution to the corresponding MBA3 instance as follows. First, we copy each of the $q$ triples in $T'$ to duties in our solution of MBA3 by selecting the corresponding triple node $t'_j$ in $V_2$, together with the associated nondummy node $x'_i$ from $V_1$ and nondummy node $z'_k$ from $V_3$ (notice that the corresponding edges are in $E$). The resulting $q$ duties contain all nondummy nodes in $V_1$ as well as all nondummy nodes in $V_3$. Further, we build duties containing the dummy nodes in $V_1$ by assigning each such node to the triple node in $V_2$ that contains element $y_j$ corresponding to the dummy node in $V_1$. This is always possible, since the instance of 3DM is a YES-instance, and hence, for each $y_j \in Y$, exactly $\#occ(y_j) - 1$ nodes in $V_2$ remain. Since the edge set $E$ contains any edge between a dummy node in $V_3$ and a node in $V_2$, we can extend these pairs to duties by assigning the dummy nodes in $V_3$ to these pairs. Observe that each resulting duty has cost 1.

We now show that a NO-instance of 3DM corresponds to a MBA3 instance which has cost 2 (or is infeasible). We do this by showing that if the constructed MBA3 instance has a solution with cost 1, then the 3DM instance is always a YES-instance. Consider a solution to the instance of MBA3 from above construction with cost 1. This means that no non-dummy node from $V_3$ is in a duty with a dummy node from $V_1$. Consider the $p - q$ duties that contain the dummy nodes of $V_1$: these duties will also contain all dummy nodes of $V_3$ (otherwise the cost of this solution would exceed 1). The remaining $q$ duties of our solution must be such that each of them contains a triple node from $V_2$, a nondummy node from $V_1$, and a nondummy node from $V_3$. These duties correspond to $q$ triples that define a solution to 3DM. □

Notice that the instances we have constructed are special in the sense that all weights are in $\{0, 1\}$. In addition, the arguments go through when the degree of each node is bounded by some constant. Based on Lemma 3, and on the preceding discussion we can now state:

**Theorem 4.** *There is no polynomial time algorithm for* MBA3 *that achieves an approximation guarantee of* $2 - \epsilon$*, for any* $\epsilon > 0$*, unless* $P = NP$*.*

Observe that the instances we constructed are such that neither $E_1$ nor $E_2$ are complete. This is a necessary property of these instances, as witnessed by the result in the next section.

## 4. A $\frac{3}{2}$-approximation algorithm for MBA3 with a single complete edge set

In this section we present a $\frac{3}{2}$-approximation algorithm for instances of MBA3 where the edge set $E_2$ (i.e., the edge set between $V_2$ and $V_3$) is arbitrary and the edge set $E_1$ (i.e., the edge set between $V_1$ and $V_2$) is complete. Notice that, for complete-MBA3, both [9, 6] present $\frac{3}{2}$-approximation algorithms. These heuristics, however, do not seem to be generalizable to MBA3 with a single complete edge set while preserving the approximation factor.

We will call our algorithm heuristic AB (Assign, then Bottleneck). The main part of AB consists of the procedure called *Core*-AB, which takes a guess of the optimal cost from the interval $\left[\left\lceil \frac{\sum_{v\in V} w(v)}{3n}\right\rceil, W\right]$, where $W$ is the largest weight occurring in the input, and finds a feasible solution with the cost at most $\frac{3}{2}$ times the guess, if one exists. AB outputs the feasible solution corresponding to the smallest guess.

**Algorithm 1** Heuristic-AB

```
{Input: MBA3 Instance}
Call Sequential Bottleneck(SB)
if no feasible solution found then
    STOP
    Output: No feasible solution
end if
$up = W$
$low = \left\lceil \frac{\sum_{v\in V} w(v)}{3n} \right\rceil$
repeat
    $p = \frac{up+low}{2}$
    Call Core-AB($p$) (see description in Section 4.1)
    if no feasible solution exists then
        $low = p$
    else
        $up = p$
    end if
until $up = low$
Call Core-AB($up$)
Output: Solution
```

In the description of *Core*-AB, we assume that the value of the optimal solution, called $OPT$, is known. This is not a problem as $OPT \in \left[\left\lceil \frac{\sum_{v\in V} w(v)}{3n}\right\rceil, W\right]$. Without loss of generality, we further assume that the nodes in $V_1, V_2$, and $V_3$ are ordered in non-increasing order of their weights, with $V_1 = \{u_1, u_2, \ldots, u_n\}$ and $V_2 = \{v_1, v_2, \ldots, v_n\}$. We say that a node $u \in V$ is *heavy* if $w(u) > \frac{OPT}{2}$, and we call $u$ *non-heavy* otherwise. Let $k_i$ be the number of heavy nodes in $V_i$, $i = 1, 2, 3$; notice that $k_1 + k_2 + k_3 \le n$.

### 4.1. The description of Core-AB

*Core*-AB has two stages. In the first stage, we solve an instance of the maximum weight perfect matching problem on a bipartite graph $G' = (V_2, V_3, E_2')$. The edge set $E_2'$ is defined as follows: there is an edge $(v, z) \in E_2'$ (with $(v, z) \in V_2 \times V_3$), if (i) $(v, z) \in E_2$, and (ii) $w(v) + w(z) \le OPT$. Further, we define the weight $w'$ of an edge $(v, z) \in E_2'$ as follows:

$$\begin{aligned} w'(v,z) &= 1 \quad \text{if } w(v) + w(z) \le OPT/2 \\ &= 0 \quad \text{otherwise} \end{aligned}$$

In the second stage, we compute a bottleneck matching (see Section 3.1) between the nodes from $V_1$ and the $n$ pairs found in Stage 1; this gives us the solution of AB.

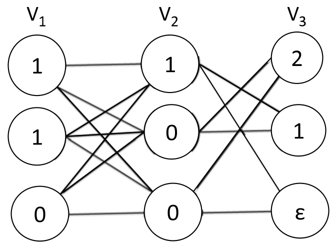

**Fig. 2.** A worst case instance for heuristic AB.

### 4.2. The analysis of heuristic AB

**Theorem 5.** *AB is a polynomial-time $\frac{3}{2}$-approximation algorithm for* MBA3 *with a single complete edge set. Moreover, there exist instances for which this bound is tight.*

**Proof.** To begin with: AB is a polynomial-time algorithm. The first stage of *Core*-AB amounts to solving a maximum weight bipartite perfect matching problem, which can be done in polynomial time and *Core*-Ab is run at most $\log W$ times.

Let us next argue that heuristic AB finds a feasible solution whenever one exists. Indeed, assuming an optimal solution exists, any pair of nodes $(v, z)$ with $(v, z) \in V_2 \times V_3$ that are together in a duty in an optimal solution, are connected in $G'$. This is true, since these $(v, z)$ apparently satisfy $(v, z) \in E_2$, and $w(v) + w(z) \le OPT$. Thus, a perfect matching exists in $G'$, and since we solve an assignment problem in the first stage of AB, we find a perfect matching in the first stage. Then it follows easily that, since $E_1$ is complete by assumption, a feasible solution is found by AB.

We now prove the approximation guarantee. Again due to the existence of an optimal solution there exists a perfect matching in $G'$ such that there are at least $k_1$ edges that are vertex disjoint whose weight in the original graph is bounded by $OPT/2$. Therefore, a maximum weight matching in $G'$ will have weight at least $k_1$. Clearly in the second stage the $k_1$ heavy elements from $V_1$ can be bottleneck matched with these pairs. The total weight of each of these triples will be bounded by $\frac{3}{2}OPT$. Any other triple will consist of three nonheavy nodes and hence its weight is also bounded by $\frac{3}{2}OPT$. This proves the approximation factor. Finally, we depict in Fig. 2 an instance for which this bound is achieved: for each $\epsilon > 0$, AB can find a solution of value 3, while $OPT = 2+\epsilon$. □

Observe that the smallest guess of the optimal cost for which *Core*-AB returns a feasible solution need not be $OPT$. For example in the instance in Fig. 2 the solution found by *Core*-AB is the same for any guess in the interval $[2, 2 + \epsilon + \alpha]$, where $0 \le \alpha < \epsilon$.

## 5. A polynomial time approximation scheme for complete-MBA

In this section we describe a polynomial time approximation scheme for complete-MBA.

First, we choose some $\epsilon$ with $0 < \epsilon \le \frac{1}{2}$ and let $W$ be the largest weight in the instance. Second, we round up each weight $w(v)$, $v \in V$ in the instance to the smallest possible multiple of $\epsilon \cdot W$. Observe that for these rounded-up weights there are $\left\lceil \frac{1}{\epsilon}\right\rceil + 1$ distinct values: $0, \epsilon \cdot W, 2\epsilon \cdot W, \ldots, \left\lceil \frac{1}{\epsilon}\right\rceil \epsilon \cdot W$; we define

$$K = \left\{0, \epsilon \cdot W, 2\epsilon \cdot W, \ldots, \left\lceil \frac{1}{\epsilon}\right\rceil \epsilon \cdot W\right\}.$$

Then, the cartesian product of $m$ sets $K$, referred to as $K^m$, is the set of all $m$-tuples $a = (a(1), a(2), \ldots, a(m))$, such that $a(j) \in K$ for each $j = 1, 2, \ldots, m$. We define, for each $j = 1, 2, \ldots, m$, and for each $k \in K, A_{jk} = \{a \in K^m : a(j) = k\}$. Further, for each $j = 1, 2, \ldots, m$, and for each $k \in K$, we define:

**Definition 6.** $p_{jk}$ equals the number of weights $w(v_i)$ with $v_i \in V_j$ that — when rounded up — equal $k\epsilon \cdot W$.

Consider now the following integer program that uses, for each $a \in K^m$, a variable $x_a$ and variable $y_a$ which takes a value 1 if $x_a$ is positive and 0 otherwise. Further, let $c_a$ denote the weight of such an $m$-tuple: $c_a = a(1) + a(2) + \cdots + a(m)$ for each $a \in K^m$ and let $M$ be a constant greater than $n$.

$$\min \; w \tag{5.1}$$

$$\text{s.t.} \quad w \geq c_a y_a \quad \forall a \in K^m, \tag{5.2}$$

$$x_a \leq M y_a \quad \forall a \in K^m, \tag{5.3}$$

$$\sum_{a \in A_{jk}} x_a = p_{jk} \quad \forall j = 1, 2, \ldots, m, k \in K, \tag{5.4}$$

$$x_a \text{ integral} \quad \forall a \in K^m. \tag{5.5}$$

$$y_a \in \{0, 1\} \quad \forall a \in K^m. \tag{5.6}$$

Observe that a solution to this integer program corresponds to a solution of complete-MBA. Indeed, a solution to the rounded-up instance of complete-MBA features $nm$-tuples such that each rounded-up weight $k\epsilon \cdot W$ of set $V_j$ occurs $p_{jk}$ times. Moreover, both the number of variables and the number of constraints are of $O(|K|^m)$, which is polynomial in the input size (for a fixed $\epsilon$). Thus we can use Lenstra's algorithm to solve this IP in polynomial time. Notice that IP's similar to (5.1)–(5.6) are used in the context of fixed parameter tractability [7]. Moreover, the core of our model, constraints (5.4) and (5.5) is in fact, nothing else than an (integral) bottleneck axial $m$-index transportation problem (see e.g. Queyranne and Spieksma [11]); it is conceivable that better algorithms than Lenstra's suffice to solve this problem exactly.

The error incurred by a solution to this IP is bounded by the error that is incurred for a single $m$-tuple (or duty), which in turn can be no more than the error induced by rounding the $m$ elements, which equals $m\epsilon W$. In other words

$$SOL \leq OPT + m\epsilon W \leq (1 + m\epsilon)OPT,$$

where SOL refers to value of the solution found by solving the integer program (5.1)–(5.6).

Notice that, given Fact 1, a PTAS for complete-MBA is the strongest result possible (unless $P = NP$).

## 6. Online algorithms for MBA

In this section we consider the online version of MBA. In the online case, the sets $V_1, \ldots, V_m$ are revealed one by one in that order. Edges between $V_i$ and $V_{i+1}$ are disclosed when $V_{i+1}$ arrives; once $V_{i+1}$ arrives, nodes of $V_{i+1}$ must be immediately and irrevocably matched with the $n$ current $i$-tuples from $V_1 \times \cdots \times V_i$. We call the online version of MBA: *online-MBA*.

Online-MBA can be seen as generalization of online parallel machine scheduling (where jobs with integer weights arrive one by one, and a job is immediately assigned to a machine upon its arrival; the objective is to minimize the makespan). Given an instance of online parallel machine scheduling with $n$ machines, we can construct an instance of online-MBA as follows: For each job $i$, create a set $V_i$ with $n - 1$ dummy nodes with weight 0, and a node with weight equal to the weight of job $i$. Each dummy node in $V_{i+1}$ has edges to every node of $V_i$, and the node with non-zero weight in $V_{i+1}$ has edges to those nodes in $V_i$ which correspond to machines to which job $i + 1$ can be assigned.

The Sequential Bottleneck Heuristic (SB), described in Section 3.1 is a natural algorithm for the online-MBA. In fact, it generalizes List Scheduling. However, in the following we show that SB can be arbitrarily bad.

**Theorem 7.** *Sequential Bottleneck Heuristic can be arbitrarily bad for online-MBA.*

**Proof.** Let $k$ be an odd integer. We construct an instance where SB finds a solution with value $k$, whereas the optimal value equals 1. We set $m := 2k - 1$ and $n := k$. Further, $V_i := \{\alpha, 0, \ldots, 0\}$ where $\alpha = 1$ if $i$ is odd, and 0 otherwise, for $i = 1, \ldots, m$. The edge set between $V_i$ and $V_{i+1}$ is complete when $i$ is odd, and when $i$ is even, the only edges are: $(u_j, v_j) \in V_i \times V_{i+1}$, for $j = 1, \ldots, n$. Observe that SB gives a solution with value $k$, whereas the optimum solution has value 1. □

## Acknowledgments

This research was supported by OT Grant OT/07/015 and carried out when the third author visited the London School of Economics (LSE). We thank Gautam Appa for stimulating discussions, and Gerhard Woeginger for pointing out references.

## References

[1] R. Aringhieri, R. Cordone, The multicommodity multilevel bottleneck assignment problem, Electronic Notes in Discrete Mathematics 17 (2004) 35–40.
[2] L. Bianco, M. Bielli, A. Mingozzi, S. Ricciardelli, M. Spadoni, A heuristic procedure for the crew rostering problem, European Journal of Operational Research 58 (1992) 272–283.
[3] H. Bodlaender, K. Jansen, G. Woeginger, Scheduling with incompatible jobs, Discrete Applied Mathematics 55 (1994) 219–232.
[4] R. Burkard, M. Dell'Amico, S. Martello, Assignment Problems, SIAM, 2009.
[5] P. Carraresi, G. Gallo, A multi-level bottleneck assignment approach to the bus drivers rostering problem, European Journal of Operational Research 16 (1984) 163–173.
[6] E.J. Coffman, M. Yannakakis, Permuting elements within columns of a matrix in order to minimize maximum row sum, Mathematics of Operations Research 9 (1984) 384–390.
[7] M.R. Fellows, S. Gaspers, F.A. Rosamond, Parameterizing by the number of numbers, in: Lecture Notes in Computer Science, vol. 6478, 2010, pp. 123–134.
[8] D. Goossens, S. Polyakovskiy, F. Spieksma, G. Woeginger, The approximability of three-dimensional assignment problems with bottleneck objective, Optimization Letters 4 (2010) 4–17.
[9] W.L. Hsu, Approximation algorithms for the assembly line balancing crew scheduling problem, Mathematics of Operations Research 9 (1984) 376–383.
[10] B. Klinz, G. Woeginger, A new efficiently solvable case of the three-dimensional axial bottleneck assignment problem, in: Lecture Notes in Computer Science, vol. 1120, 1996, pp. 150–162.
[11] M. Queyranne, F. Spieksma, Approximation algorithms for multi-index transportation problems with decomposable costs, Discrete Applied Mathematics 76 (1997) 239–254.