DEPARTMENT OF OPERATIONS RESEARCH
UNIVERSITY OF AARHUS

Working Paper no. 2005 / 3 2005 / 12 / 27

On the Bicriterion Multi Modal Assignment Problem

Christian Roed Pedersen, Lars Relund Nielsen
and Kim Allan Andersen

ISSN 1600-8987

Department of Mathematical Sciences
Telephone: +45 8942 1111
E-mail: institut@imf.au.dk

Building 530, Ny Munkegade
DK-8000 Aarhus C, Denmark
URL: www.imf.au.dk

On the Bicriterion Multi Modal Assignment Problem

Christian Roed Pedersen

Department of Operations Research

University of Aarhus

Ny Munkegade, Building 1530

8000 Aarhus C

Denmark

Lars Relund Nielsen Kim Allan Andersen*

Department of Accounting, Finance and Logistics

Aarhus School of Business

Fuglesangs Allé 4

DK-8210 Aarhus V

Denmark

December 22, 2005

Abstract

We consider the bicriterion multi modal assignment problem which is a new generalization of the classical linear assignment problem. A two-phase solution method using an effective ranking scheme is presented. The algorithm is valid for generating all nondominated criterion points or an approximation.

Extensive computational results are conducted on a large library of test instances to test the performance of the algorithm and to identify hard test instances.

Also, test results of the algorithm applied to the bicriterion assignment problem is given. Here our algorithm outperforms all previously known exact solution methods for this problem class.

Keywords: Bicriterion multi modal assignment problem, ranking, two-phase method.

*Corresponding author, e-mail: kia@asb.dk.

1 Introduction

The *linear assignment problem* (AP) is a well-known combinatorial problem with applications in a widespread field of operations. In its most classical formulation, AP is described as the problem of assigning n workers to n jobs such that cost is minimized.

In 1955 Kuhn [12, 13] presented the first polynomial solution method for AP, called the *Hungarian method*. Later, alternative AP algorithms have been presented among which some of the most efficient are the *successive shortest path algorithms*[1], see e.g. Ahuja, Magnanti, and Orlin [1] and Jonker and Volgenant [11]. Dell'Amico and Martello [5] give an annotated bibliography on AP, mentioning more than 100 papers. An excellent survey is given by Dell'Amico and Toth [6] including comparative tests of several implementations of AP algorithms.

Different generalizations of AP have generated interest in the literature. One prominent research field considers ranking the K best assignments in nondecreasing order of cost. Applications for ranking problems are numerous. For instance, they are often used with success as subroutines for more complex optimization problems. Recent developments have been given by Pascoal, Captivo, and Clímaco [18] and Pedersen, Nielsen, and Andersen [19].

In general a description of real world applications as single criterion optimization problems is seldom realistic, since they are often by nature imposed with more objectives to be simultaneously optimized. Assigning workers to jobs with minimal cost and time yields the *bicriterion assignment problem* (BiAP), which is another important generalization of AP. Within the last ten years focus on BiAP has risen. Ulungu and Teghem [24] presented the first exact solution method for BiAP, proposing a two-phase method identifying in phase one all supported efficient solutions and in phase two all unsupported efficient solutions. In that paper, a scheme resembling total enumeration in all nonbasic variables is employed. The method in [24] was implemented by Tuyttens, Teghem, Fortemps, and Nieuwenhuyze [23], showing – with large CPU times – the limitations of this algorithm. Recently, an improvement of this algorithm was given in Przybylski, Gandibleux, and Ehrgott [20], proposing also a two-phase method applying ranking for BiAP.

The main focus on BiAP in literature, however, seems to be on heuristical methods. Tuyttens et al. [23] use a version of the MOSA method which is an extension of simulated annealing to deal with multiple objectives. Gandibleux, Morita, and Katoh [8] use genetic information for BiAP, and population based heuristics using path relinking are described in Gandibleux, Morita, and Katoh [9] and Przybylski et al. [20].

In this paper, we deal with another highly relevant extension of the classical assignment problem, which has, to the best of our knowledge, not yet been discussed in literature. Imagine a large global company with n specialists spread across the world and suppose that exactly n jobs have to be performed by these n specialists. Hence, each specialist must be assigned to exactly one job and furthermore suitable modes of transportation must be chosen for the workers to travel to the destinations of the jobs. For this problem, it seems relevant to consider two weight criteria to be

[1] Also known as shortest augmented path algorithms.

minimized simultaneously, namely *travel time* and *travel or assignment cost*. Since a specialist i has possibly multiple modes of transportation and routes to choose from in order to reach the destination of job j, several two-dimensional cost vectors exist for each i and j. Therefore, the *bicriterion multi modal assignment problem* (BiMMAP) is an extension of BiAP containing, in each assignment cell, several two-dimensional cost vectors/points. The objective is to identify either all efficient assignments or all nondominated criterion points for the problem. Notice that the predominant thought within bicriterion optimization is to identify all nondominated points with one efficient solution corresponding to each nondominated point. This is equivalent to identifying a *minimal complete set of efficient solutions* [9, 10].

In this paper, we propose a two-phase method to identify all the nondominated criterion points for BiMMAP. Acknowledging that ranking procedures have been applied with great success for other bicriterion optimization problems (see e.g. Nielsen, Andersen, and Pretolani [16]), we employ ranking of multi modal assignments as a subroutine. The subroutine is an efficient extension of the algorithm for finding the K best assignments by Pedersen et al. [19].

A method for finding an ε-approximation (Warburton [25]) of the set of nondominated points, which enable us to control the quality of the set of criterion points reported, is also presented. An approximation may be needed for large problem sizes if it is too time-consuming to find all nondominated criterion points.

Using a large library of test instances for BiMMAP, we give numerical results indicating the effectiveness of our method. The concept of approximating the nondominated points is shown to have a large effect on the computational performance. Since BiMMAP is a generalization of BiAP, we also report computational results for some BiAP instances previously solved in literature showing that our algorithm outperforms all known exact solution methods.

The paper is organized as follows. In Section 2 we introduce the bicriterion multi modal assignment problem and give a few theoretical results for this problem class. In Section 3 we describe our two-phase method both for the exact and the approximation solution method. Section 4 provides a description on how to rank multi modal assignments. Computational results for BiMMAP and BiAP are given in Section 5, and conclusions are drawn in Section 6.

2 The bicriterion multi modal assignment problem

In this section we give the mathematical formulation of *the bicriterion multi modal assignment problem* (BiMMAP), introduce the relevant terminology and give a few theoretical results.

Imagine a large global company with n specialists spread across the world and suppose that exactly n jobs have to be performed by these n specialists. That is, each specialist must be assigned to exactly one job. Moreover, a specialist i has L_{ij} different mode choices of transportation for reaching the destination of job j with *travel or assignment cost* c^1_{ijl} and *travel time* c^2_{ijl}, $l = 1, \ldots, L_{ij}$.

BiMMAP is an extension of BiAP where, for each cell (i, j) in the *assignment*

$i\backslash j$	1	$\cdots$	n
1	$\binom{c^1_{111}}{c^2_{111}}, \ldots, \binom{c^1_{11L_{11}}}{c^2_{11L_{11}}}$	$\cdots$	$\binom{c^1_{1n1}}{c^2_{1n1}}, \ldots, \binom{c^1_{1nL_{1n}}}{c^2_{1nL_{1n}}}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
n	$\binom{c^1_{n11}}{c^2_{n11}}, \ldots, \binom{c^1_{n1L_{n1}}}{c^2_{n1L_{n1}}}$	$\cdots$	$\binom{c^1_{nn1}}{c^2_{nn1}}, \ldots, \binom{c^1_{nnL_{nn}}}{c^2_{nnL_{nn}}}$

Figure 1: The cost matrix of BiMMAP.

cost matrix, we have several two-dimensional cost vectors as illustrated in Figure 1. The objective is to identify either all efficient minimal cost assignments or all non-dominated criterion points for the problem.

Let x_{ijl} be a binary variable with value 1, if i is assigned to j using mode choice l, and 0 otherwise. Obviously, exactly one i must be assigned to each j using a specific mode choice which gives us the following mathematical formulation of BiMMAP.

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{L_{ij}} c^1_{ijl}x_{ijl} \\
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{L_{ij}} c^2_{ijl}x_{ijl} \\
\text{st.} \quad & \sum_{j=1}^{n}\sum_{l=1}^{L_{ij}} x_{ijl} = 1, \;\; i = 1, 2, \ldots, n \\
& \sum_{i=1}^{n}\sum_{l=1}^{L_{ij}} x_{ijl} = 1, \;\; j = 1, 2, \ldots, n \\
& x_{ijl} \in \{0, 1\} \;\; \forall i, j, l
\end{aligned} \tag{1}$$

We assume that for each cell (i, j) the costs satisfy

$$0 < c^1_{ij1} < c^1_{ij2} < \cdots < c^1_{ijL_{ij}} \text{ and } c^2_{ij1} > c^2_{ij2} > \cdots > c^2_{ijL_{ij}} > 0 \tag{2}$$

Moreover, c^1_{ijl} and c^2_{ijl} are integer for all i, j and l.

A feasible solution x to (1) is called a *multi modal assignment* or short an *assignment*. An assignment may alternatively be written as $a = \{(1, j_1, l_1), \ldots, (n, j_n, l_n)\}$ where $(i, j, l) \in a$ if and only if $x_{ijl} = 1$.

Let the *multi modal assignment polytope* $\mathcal{MMAP}$ be the set of all feasible solutions to the continuous relaxation of (1). For $x \in \mathcal{MMAP}$, we denote by $y = (y_1, y_2) = (c^1x, c^2x)$ the corresponding objective vector (or criterion vector). Note that if $L_{ij} = 1$ for all i, j, $\mathcal{MMAP}$ reduces to the assignment polytope, $\mathcal{AP}$.

2.1 Methodology

For single criterion optimization, the concept of optimality is well-defined. However, minimizing a vector-valued objective function requires some more explanation since there is no complete order defined in $\mathbb{R}^p$ for $p \geq 2$. Respecting common practice in the field of multicriteria optimization, we shall deploy the Pareto concept of optimality, which is based on the following binary relation. Let $y^1, y^2 \in \mathbb{R}^2$. Then

$$y^1 \leq y^2 \quad \Leftrightarrow \quad y^1_r \leqq y^2_r \quad r = 1, 2 \quad \textit{and} \quad y^1 \neq y^2$$

A point $y^2 \in \mathbb{R}^2$ is *dominated* by $y^1 \in \mathbb{R}^2$ if $y^1 \leq y^2$.

Consider the following biobjective minimization problem:

$$\begin{aligned} &\min \ c^1 x \\ &\min \ c^2 x \\ &\text{s.t.} \ \ x \in \mathcal{X} \end{aligned} \tag{3}$$

where $\mathcal{X}$ denotes the set of feasible solutions also referred to as the *decision space*. Let $\mathcal{Y} = \{(y_1, y_2) \in \mathbb{R}^2 \mid y_1 = c^1 x, \ y_2 = c^2 x, \ x \in \mathcal{X}\}$ denote the corresponding *criterion space*. The *efficient* set $\mathcal{X}_E$ is defined as

$$\mathcal{X}_E = \{x \in \mathcal{X} \mid \nexists \bar{x} \in \mathcal{X} : (c^1 \bar{x}, c^2 \bar{x}) \text{ dominates } (c^1 x, c^2 x)\},$$

and the *nondominated set* $\mathcal{Y}_N$ is given by

$$\mathcal{Y}_N = \{(y_1, y_2) \in \mathbb{R}^2 \mid y_1 = c^1 x, \ y_2 = c^2 x, \ x \in \mathcal{X}_E\}.$$

The nondominated points in $\mathcal{Y}_N$ can be partitioned into *supported* and *unsupported* points. The supported ones can be further subdivided into *extreme* and *nonextreme*. To this aim, let us define the following set

$$\mathcal{Y}^{\geq} = \text{conv}\left\{\mathcal{Y}_N \oplus \{y \in \mathbb{R}^2 : y \geq 0\}\right\}$$

where $\oplus$ denotes the usual direct sum. A point $y \in \mathcal{Y}_N$ is a *supported* nondominated point if it is on the boundary of $\mathcal{Y}^{\geq}$; otherwise it is *unsupported*. A supported nondominated point y is *extreme* if it is an extreme point of $\mathcal{Y}^{\geq}$; otherwise it is *nonextreme*.

The criterion space is illustrated in Figure 2. Nondominated points are displayed as dots and $\mathcal{Y}^{\geq}$ is the shaded area. Supported points are y^1-y^5 of which y^3 is the only nonextreme. Unsupported nondominated points are y^6 and y^7 while y^8 is dominated.

In some cases it may be enough to find an approximation of the nondominated set. In this case we need the concepts of ε-domination and ε-approximation introduced by Warburton [25]. A point $y = (y_1, y_2)$ *ε-dominates* point $\hat{y} = (\hat{y}_1, \hat{y}_2)$ if $(1 - \varepsilon)y$ dominates $\hat{y}$. A set $\mathcal{Y}_1$ is an *ε-approximation* of a nondominated set $\mathcal{Y}_2$, if, for each point $\hat{y} \in \mathcal{Y}_2$, there exists $y \in \mathcal{Y}_1$ that ε-dominates it.

It is well-known that unsupported nondominated points may exist for BiAP [7], and hence also for BiMMAP. Also, due to the fact that BiMMAP is a generalization of BiAP, it holds true that BiMMAP is intractable and $\mathcal{NP}$-complete ([7, 22]). Moreover, because of (2), we have that all cost vectors for a cell are nondominated.

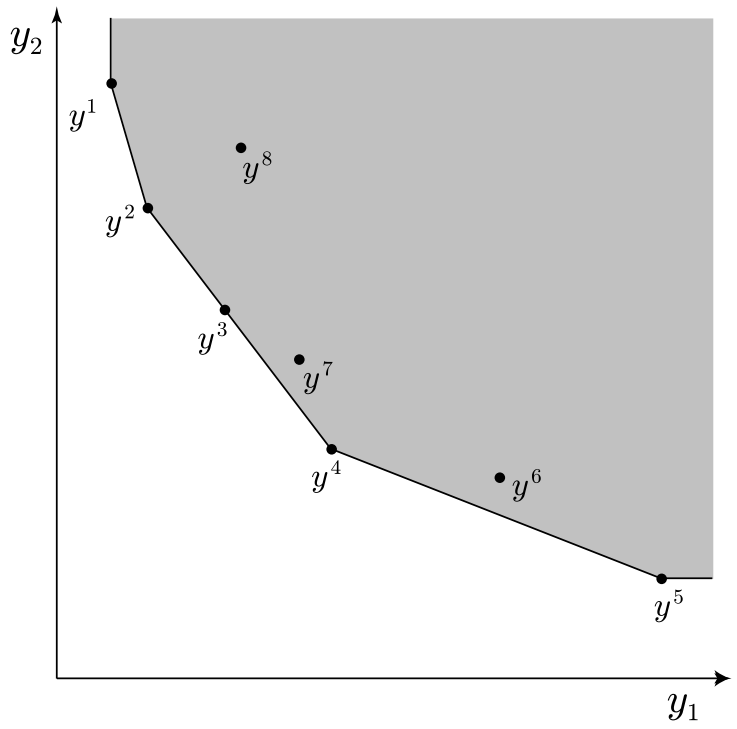


Figure 2: The criterion space.

This is no restriction, since a dominated cost vector in a given cell will never be used in an efficient assignment.

Let $G = (V, E)$ denote the *adjacency graph* of $\mathcal{MMAP}$, where V is the set of efficient basic feasible solutions to the continuous relaxation of (1). An edge between two nodes of V is included in E, if and only if the corresponding efficient basic feasible solutions can be obtained from each other by a single pivot operation. Along the same lines as in Przybylski, Gandibleux, and Ehrgott [21], it can be shown that the adjacency graph for $\mathcal{MMAP}$ may not be connected. In particular, this means that it may not be possible to find the full set of nondominated solutions by simple pivot operations. Therefore, to find such a full set of nondominated points, we propose to use a two-phase method not based upon simplex operations. We explain our two-phase in the next section.

3 Solving BiMMAP using the two-phase method

The two-phase approach is a general method for solving bicriterion combinatorial problems such as (3). As the name suggests, the two-phase method divides the search for nondominated points into two phases.

In phase one, the supported extreme nondominated points are found. These extreme points define a number of triangles in which unsupported nondominated points may be found. Phase two proceeds to search the triangles one at a time. Both phases make use of a parametric minimization problem defined as follows:

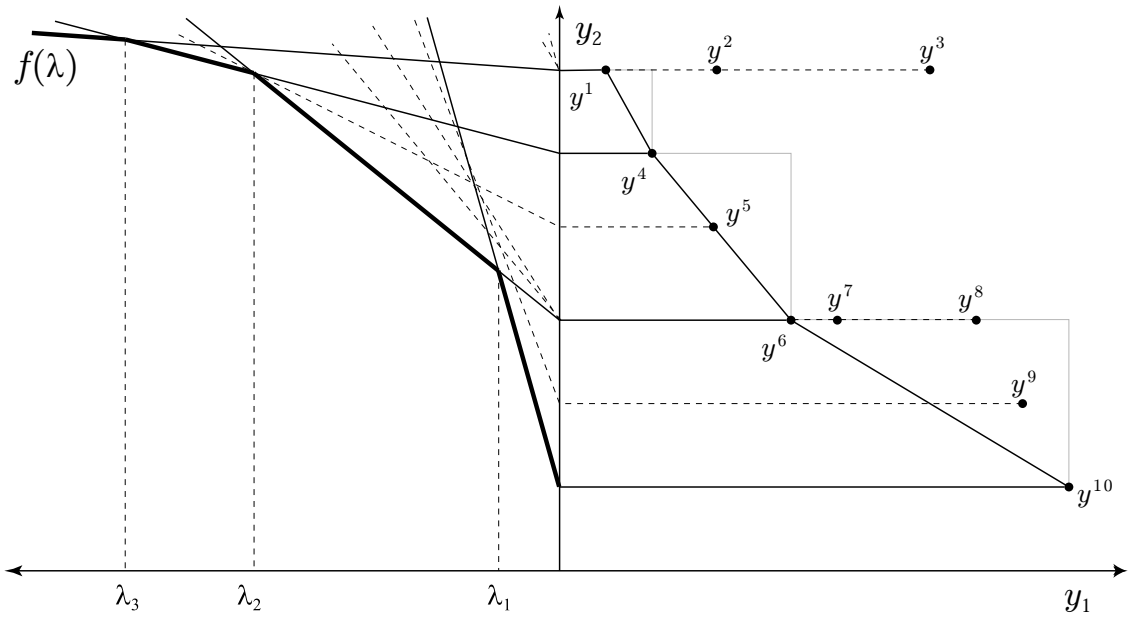


Figure 3: The criterion space and its corresponding parametric space.

$$\begin{array}{ll} \min & f_\lambda(x) = (\lambda c^1 + c^2)x \\ \text{s.t.} & x \in \mathcal{X} \end{array} \tag{4}$$

The method is best illustrated using an example. Suppose that the points in the right side of Figure 3 represent the criterion space of the biobjective minimization problem (3). Points y^1, y^4, y^5, y^6 and y^{10} are supported nondominated points of which y^5 is the only nonextreme. y^9 is the only unsupported nondominated point. The remaining points are dominated.

Consider the parametric problem (4). For a fixed criterion point $y = (c^1x, c^2x)$, $f_\lambda(x)$ define a line with slope $y_1 = c^1x$ and intersection $y_2 = c^2x$ in the *parametric space* as illustrated on the left side of Figure 3. The lower envelope of the lines in the parametric space defines a non-decreasing piecewise linear function $f(\lambda)$ with break points λ_i. Note that each line on $f(\lambda)$ corresponds to an extreme nondominated point. As a result, each extreme nondominated point can be found by identifying the point with minimal parametric weight for fixed λ values, i.e. solving (4). This is done in phase one which uses a NISE[2] like algorithm (see [3]) as shown in Figure 4. This idea was first applied to the bicriterion transportation problem by Aneja and Nair [2].

The procedure first finds the *upper/left* and the *lower/right* point (y^1 and y^{10} in Figure 3). Given two extreme nondominated points y^+ and y^-, we calculate the *search direction* λ defined by the slope of the line between the points and solve (4). That is, we find the value of λ where the two lines corresponding to y^+ and y^- meet in the parametric space. If the optimal solution $x^\star$ of (4) corresponds to a

[2]Non-inferior set estimation.

```
1  procedure PhaseOne()
2     y^UL := (c^1 x^UL, c^2 x^UL), where x^UL is optimal for lexmin(c^1 x, c^2 x);
3     y^LR := (c^1 x^LR, c^2 x^LR), where x^LR is optimal for lexmin(c^2 x, c^1 x);
4     if (y^UL = y^LR) then STOP (only one nondominated point);
5     Y := {y^UL, y^LR};
6     y^+ := y^UL;  y^- := y^LR;
7     while (y^+ ≠ y^LR) do
8        λ := (y_2^+ − y_2^−)/(y_1^− − y_1^+);
9        solve (4) with optimal decision x* and cost y* = (c^1 x*, c^2 x*);
10       if (f_λ(x*) < y_1^+ λ + y_2^+) then add y* between y^+ and y^− in Y;
11       else y^+ := y^−;
12       y^− := Next(Y, y^+);
13    end while
14 end procedure
```

Figure 4: Phase one - Finding supported extreme nondominated points.

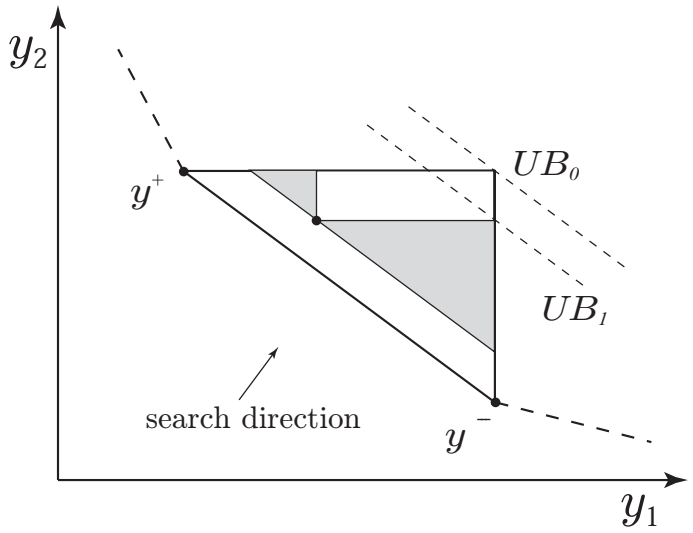


Figure 5: A triangle defined by y^+ and y^-.

new extreme nondominated point, then the parametric weight $f_\lambda(x^\star)$ must be below the parametric weight of y^+ and y^- (line 10). The points $y^+, y^\star$ and y^- then define two new search directions and the **while** step is repeated on the points y^+ and $y^\star$. Otherwise no new extreme nondominated point has been found and we proceed with the two next points in $\mathcal{Y}$, i.e we call the `Next` function that returns the point following y in $\mathcal{Y}$. The procedure stops when no additional extreme nondominated points can be found.

Since there may exist unsupported nondominated criterion points, it is not in general possible to find all nondominated points during the first phase. This can be seen in Figure 3 where unsupported nondominated points inside the triangles, such as y^9, correspond to a dashed line lying above $f(\lambda)$. These points are found in phase two which searches each triangle defined by the set of extreme nondominated points

```
1  procedure PhaseTwo(△(y⁺, y⁻))
2     λ:=(y₂⁺ − y₂⁻)/(y₁⁻ − y₁⁺);
3     𝒴:={y⁺, y⁻};
4     k:=1; LB:=λy₁⁺ + y₂⁺; UB:=UpdateUB(𝒴);
5     while (LB ≤ UB) do
6        y^k:=KBest(k, λ);
7        if (NonDom(y^k)) then
8           𝒴:=𝒴 ∪ {y^k};
9           UB:=UpdateUB(𝒴);
10       end if
11       LB:=λy₁^k + y₂^k; k:=k + 1;
12    end while
13 end procedure
```

Figure 6: Phase two - Finding unsupported extreme nondominated points.

found in phase one.

Consider the triangle $\triangle(y^+, y^-)$ defined by the extreme nondominated points y^+ and y^- (see Figure 5). The second phase searches each triangle using a K best procedure to rank the parametric weight $f_\lambda(x)$ in the search direction defined by the slope between the two points defining the triangle. The search stops when the parametric value $f_\lambda(x)$ reaches an upper bound. Initially, the upper bound is $UB_0 = y_1^-\lambda + y_2^+$. When a new unsupported nondominated point is found inside the triangle, the upper bound is updated to UB_1 as can be seen in Figure 5.

A pseudo code is given in Figure 6 where initialization is done on lines 2-4. In the main loop the parametric weight $f_\lambda(x)$ is ranked until the upper bound is reached. Procedure `KBest` returns the cost vector y^k of the k'th best solution. If it is nondominated, we add y^k to the nondominated set and update the upper bound using `UpdateUB`. Finally, we update the lower bound to the parametric weight and repeat the loop.

A detailed description of procedure `KBest` is given in Section 4. Furthermore, note that in general `UpdateUB` finds the upper bound by using the points in the nondominated set. However, special properties such as integrality of the nondominated points may be used to improve the bound, as we will see in the following sections.

3.1 BiMMAP – Finding the complete set of nondominated points

To solve BiMMAP using the two-phase method, we must be able to solve the parametric problem:

$$\begin{aligned} &\min \ f_\lambda(x) = \left(\lambda c^1 + c^2\right) x = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{L_{ij}} (\lambda c_{ijl}^1 + c_{ijl}^2) x_{ijl} \\ &\text{s.t.} \quad x \in \mathcal{MMAP} \end{aligned} \tag{5}$$

For a given value of λ and a given cell (i,j), define

$$l^\star(\lambda) = \text{argmin}_{1\le l\le L_{ij}}\{\lambda c^1_{ijl} + c^2_{ijl}\} \tag{6}$$

That is, for a specific value of λ the minimal parametric cost entry, $l^\star$, in each assignment cell (i,j) is chosen. It is straightforward to see that (5) reduces to the following problem:

$$\begin{aligned} &\min\ f_\lambda(x) = \left(\lambda c^1 + c^2\right) x = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijl^\star(\lambda)} x_{ijl^\star(\lambda)} \\ &\text{s.t.}\quad x \in \mathcal{AP} \end{aligned} \tag{7}$$

which is a single criterion assignment problem. This is summarized in Proposition 1.

Proposition 1. *The parametric problem* $\min\{f_\lambda(x)|x \in \mathcal{MMAP}\}$ *reduces to the classical AP* $\min\{\sum_{ij} c_{ijl^\star(\lambda)} x_{ijl^\star(\lambda)}|x \in \mathcal{AP}\}$.

Therefore, the minimal cost assignment for BiMMAP given a fixed value of λ can be found in procedure `PhaseOne` by solving a single criterion AP. Moreover, since each criterion point in BiMMAP is integer, the following obvious proposition may be used to reduce the computational effort in phase one.

Proposition 2. *Consider two extreme nondominated points* y^+ *and* y^-. *Then no further nondominated extreme points can be found below the line connecting* y^+ *and* y^- *in phase one, if any of the following conditions are fulfilled*

$$y^-_1 - y^+_1 = 1 \quad \text{or} \quad y^+_2 - y^-_2 = 1 \tag{8}$$

That is, we simply skip the search between y^+ and y^- if (8) holds in phase one. Condition (8) may also be used in phase two to skip searching some triangles. If any of the conditions in (8) are fulfilled, we do not have to apply procedure `PhaseTwo` to triangle $\triangle(y^+, y^-)$ since no unsupported nondominated point can exist in this triangle.

Observe that this also holds true even if the points y^+ and y^- are supported nonextreme nondominated points. Hence, storing such nonextreme points in phase one (by using a $\le$ sign instead of a $<$ sign on line 10 in Figure 4) may reduce the computational effort in phase two. Furthermore, since all nondominated points have integer coordinates, the upper bound used in phase two may be improved.

Proposition 3. *Given the triangle* $\triangle(y^+, y^-)$ *with previously found nondominated points* $\{y^+ = y^1, \ldots, y^q = y^-\}$ *ordered in increasing order of the first objective and a search direction given by* λ, *define*

$$UB^{IP} = \max_{i=1,\ldots,q-1}\{\lambda\left(y^{i+1}_1 - 1\right) + \left(y^i_2 - 1\right)\}. \tag{9}$$

Then all unsupported nondominated criterion points in $\triangle(y^+, y^-)$ *have parametric weight below or equal to* UB^{IP}.

Proof. Consider a non-found nondominated point (y_1, y_2) located in $\triangle(y^+, y^-)$. Due to integrality of y_1 and y_2 we have

$$\begin{aligned} & \exists i \in \{1, \ldots, q-1\}: \;\; y_1 \leq y_1^{i+1} - 1 \;\; \wedge \;\; y_2 \leq y_2^i - 1 \\ \Downarrow & \\ & \lambda y_1 + y_2 \leq \lambda\big(y_1^{i+1} - 1\big) + \big(y_2^i - 1\big) \end{aligned}$$

Since nondominated points can be located between any two consecutive points y^i and y^{i+1}, we obtain expression (9) for the upper bound. □

As a result, we can use (9) in function `UpdateUB` of procedure `PhaseTwo`. Furthermore, note that upper bound (9) is valid for all bicriterion problems with integer criterion points and is an improvement to the upper bound previously reported in literature [23, 24].

3.2 BiMMAP - Finding an approximation

In some cases it may be sufficient to find an approximation of the nondominated set. In this section we consider the problem of finding an ε-approximation ($\varepsilon > 0$) of the nondominated set, which enables us to control the quality of the approximation reported. Only slightly modified versions of phase one and two are needed.

First, consider phase one and a set of extreme nondominated points found during phase one as illustrated in Figure 7. Note that any new extreme nondominated point between y^+ and y^- must belong to the shaded area in Figure 7. As a result we can skip searching for new extreme nondominated points between y^+ and y^- if the following proposition is satisfied.

Proposition 4. *Given extreme points $\{y^{UL}, \ldots, y^s, y^+, y^-, y^t, \ldots, y^{LR}\}$ found during phase one, each extreme nondominated point between y^+ and y^-, i.e. inside the shaded area of Figure 7, is ε-dominated by either y^+ or y^- if*

$$\begin{aligned} & \lambda_1 \big\lfloor (1-\varepsilon)\, y_1^- \big\rfloor + \big\lfloor (1-\varepsilon)\, y_2^+ \big\rfloor \leq \lambda_1 y_1^+ + y_2^+ \\ \textit{or} \qquad & \lambda_2 \big\lfloor (1-\varepsilon)\, y_1^- \big\rfloor + \big\lfloor (1-\varepsilon)\, y_2^+ \big\rfloor \leq \lambda_2 y_1^- + y_2^- \end{aligned} \tag{10}$$

where λ_1 is defined by the slope of the line between y^s and y^+ and λ_2 is defined by the slope of the line between y^- and y^t.

Proof. We only show the result if the first condition in (10) is satisfied. The second case is shown with similar arguments. Therefore, assume that $\lambda_1 \big\lfloor (1-\varepsilon)\, y_1^- \big\rfloor + \big\lfloor (1-\varepsilon)\, y_2^+ \big\rfloor \leq \lambda_1 y_1^+ + y_2^+$. Furthermore, suppose that there exists a nondominated point (y_1, y_2) in the shaded area between y^+ and y^- (see Figure 7), which is not ϵ-dominated by neither y^+ nor y^-. It follows that $(1-\varepsilon)\, y_1^- > y_1$ and $(1-\varepsilon)\, y_2^+ > y_2$. Due to integrality of y_1 and y_2, this means:

$$\lambda_1 y_1 + y_2 \leq \lambda_1 \big\lfloor (1-\varepsilon)\, y_1^- \big\rfloor + \big\lfloor (1-\varepsilon)\, y_2^+ \big\rfloor \tag{11}$$

The nondominated point (y_1, y_2) is necessarily strictly above the line through y^s and y^+ (otherwise y^+ is not an extreme point found before (y_1, y_2)). Therefore, $\lambda_1 y_1 + y_2 > \lambda_1 y_1^+ + y_2^+$. This implies that

$$\lambda_1 \big\lfloor (1-\varepsilon)\, y_1^- \big\rfloor + \big\lfloor (1-\varepsilon)\, y_2^+ \big\rfloor \leq \lambda_1 y_1^+ + y_2^+ < \lambda_1 y_1 + y_2 \tag{12}$$

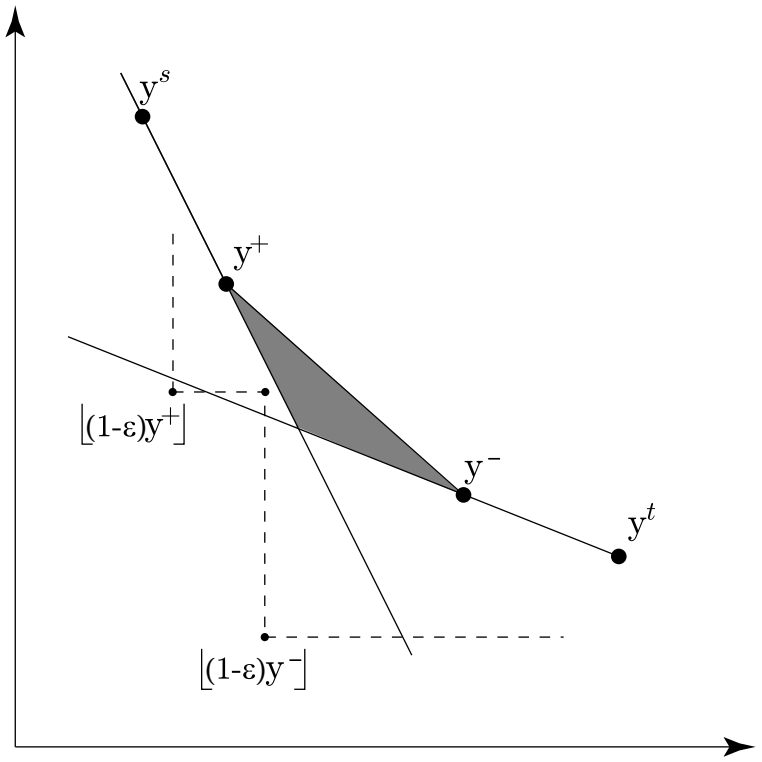


Figure 7: Using ε-dominance in the first phase.

Combining equations (11) and (12) a contradiction is obtained. □

Observe that Proposition 4 also holds true for the case where no points are identified to the left (or to the right) of the points y^+ and y^- by an appropriate choice of λ_i, $i = 1, 2$.

Corollary 1. *If $y^+ = y^{UL}$ ($y^- = y^{LR}$) Proposition 4 holds true by choosing $\lambda_1 = \infty$ ($\lambda_2 = 0$).*

Proposition 4 and Corollary 1 are used in procedure `PhaseOne` to skip the search between two points y^+ and y^-.

In phase two, the upper bound (9) can be further strengthened if an ε-approximation is wanted.

Proposition 5. *Given the triangle $\triangle(y^+, y^-)$ with previously found nondominated points $\{y^+ = y^1, \ldots, y^q = y^-\}$ ordered in increasing order of the first objective, define*

$$UB^{IP}(\varepsilon) = \max_{i=1,\ldots,q-1} \{\lambda \lfloor (1-\varepsilon) y_1^{i+1} \rfloor + \lfloor (1-\varepsilon) y_2^i \rfloor\} \tag{13}$$

Then all criterion points in $\triangle(y^+, y^-)$ with parametric weight above $UB^{IP}(\varepsilon)$ are ε-dominated by the current ε-approximation of the triangle.

Proof. Let $y = (y_1, y_2)$ be a nondominated point in $\triangle(y^+, y^-)$. Assume that y is located between y^i and y^{i+1}. If y is not ϵ-dominated then

$$
\begin{array}{ll}
 & \exists i \in \{1,\ldots,q-1\} : y_1 < (1-\varepsilon)\, y_1^{i+1} \ \ \wedge \ \ y_2 < (1-\varepsilon)\, y_2^{i} \\
\Downarrow & \text{(since } y \text{ is integral)} \\
 & y_1 \leq \left\lfloor (1-\varepsilon)\, y_1^{i+1} \right\rfloor \ \ \wedge \ \ y_2 \leq \left\lfloor (1-\varepsilon)\, y_2^{i} \right\rfloor \\
\Downarrow & \\
 & \lambda y_1 + y_2 \leq \lambda \lfloor (1-\varepsilon)\, y_1^{i+1} \rfloor + \lfloor (1-\varepsilon)\, y_2^{i} \rfloor
\end{array}
$$

Since nondominated points can be located between any two consecutive points y^i and y^{i+1}, we obtain expression (13) for the upper bound. □

It follows that for $\varepsilon > 0$ equation (13) can be used to find the upper bound in function `UpdateUB` of procedure `PhaseTwo`. Also, note that if we consider the shaded area between y^+ and y^- (see Figure 7) not searched in phase one (i.e. satisfying (10)), then $UB^{IP}(\varepsilon)$ for $\triangle(y^+, y^-)$ will be less than the parametric weight of y^+. Therefore, $\triangle(y^+, y^-)$ is not searched in procedure `PhaseTwo` either. Furthermore, there can be some possible gains in storing nonextreme supported points in phase one as well. The more supported nondominated points that are identified in the first phase, the more likely are $UB^{IP}(\varepsilon)$ to be below the parametric weight of y^+ and hence $\triangle(y^+, y^-)$ is not searched.

Note that, the approximation found in phase one is a subset of the supported nondominated points, since the optimal solution of (5) corresponds to an extreme nondominated point. Moreover, because we apply a ranking procedure in the second phase, a dominated point cannot be found before a point dominating it. These comments provide us with the following result.

Proposition 6. *The approximation of the nondominated set for an $\varepsilon > 0$ is a subset of $\mathcal{Y}_N$.*

4 Finding the K best multi modal assignments

In this section, we describe our method for ranking multi modal assignments in nondecreasing order of cost, used in phase two when searching a triangle. That is, ranking assignments using the single criterion parametric costs defined for a given parameter λ.

Without loss of generality, assume that each cell (i, j) contains entries $c_{ij1} \leq \ldots \leq c_{ijL_{ij}}$. Our objective is to determine the K best assignments $a_1, a_2, \ldots, a_K$, in a single criterion multi modal assignment problem, such that

- $c(a_i) \leq c(a_{i+1}),\ i = 1, 2, \ldots, K-1$
- $c(a_K) \leq c(a),$ for any assignment $a \notin \{a_1, \ldots, a_K\}$

where $c(a)$ denotes the cost of assignment a.

In general, ranking algorithms use a specific branching technique to partition the set of possible solutions into smaller subsets and a solution technique to find the optimal solution for each subset.

Let $\mathcal{A}$ denote the set of possible multi modal assignments. In this paper, we use a branching technique which is an extension of the branching technique originally

proposed by Murty [14]. Here we partition the set $\mathcal{A}$ into smaller subsets as follows: Given the optimal assignment $a_1 = \{(1, j_1, l_1), (2, j_2, l_2), \ldots, (n, j_n, l_n)\}$ of $\mathcal{A}$, the set $\mathcal{A} \setminus \{a_1\}$ is partitioned into $n-1$ disjoint subsets $\mathcal{A}^i$, $i = 1, \ldots, n-1$, where

$$\mathcal{A}^i = \{a \in \mathcal{A} \mid (1, j_1, l_1), \ldots, (i-1, j_{i-1}, l_{i-1}) \in a, (i, j_i, l_i) \notin a\},\ i = 1, \ldots, n-1$$

Clearly, the second best assignment a_2 can be identified using a solution technique to find the minimal cost assignment in the sets $\mathcal{A}^i$, $i = 1, \ldots, n-1$. Moreover, the branching technique can be applied recursively to subsets $\mathcal{A}^i \subset \mathcal{A}$.

In general, the algorithm maintains a candidate set Φ of pairs $(\hat{a}, \hat{\mathcal{A}})$, where $\hat{a}$ is the minimum cost assignment in subset $\hat{\mathcal{A}}$. Suppose we have found the $k-1$ best assignments $a_1, \ldots, a_{k-1}$, then the current candidate set Φ represents a disjoint partition of $\mathcal{A} \setminus \{a_1, \ldots, a_{k-1}\}$. The kth best assignment is then found as the pair $(\hat{a}, \hat{\mathcal{A}}) \in \Phi$ which contains the assignment $\hat{a}$ with minimum cost $c(\hat{a})$ among all assignments in the candidate set Φ.

Next let us consider the solution technique, i.e how to determine the minimum cost assignments in $\hat{\mathcal{A}}^i$ when applying the branching technique to some subset $\hat{\mathcal{A}} \subset \mathcal{A}$. Without loss of generality, assume that the minimum cost assignment in subset $\hat{\mathcal{A}}$ is given by

$$\hat{a} = \{(1, j_1, l_1), (2, j_2, l_2), \ldots, (n, j_n, l_n)\}. \tag{14}$$

Furthermore, assume that, according to previous partitions, no assignments in $\hat{\mathcal{A}}$ can contain $(m_1, p_1, h_1), \ldots, (m_q, p_q, h_q)$. Recall that any assignment belonging to $\hat{\mathcal{A}}^i$ must contain $(1, j_1, l_1), \ldots, (i-1, j_{i-1}, l_{i-1})$. Assuming that $\hat{\mathcal{A}}^i$ contains an assignment, it can be found as follows:

1. Delete rows $\{1, 2, \ldots, (i-1)\}$ and columns $\{j_1, j_2, \ldots, j_{i-1}\}$ from the cost matrix.

2. The cost of entries (i, j_i, l_i) and $(m_1, p_1, h_1), \ldots, (m_q, p_q, h_q)$ in the cost matrix is set to infinity.

Given a nonempty subset $\hat{\mathcal{A}}^i$, let MMAP($\hat{\mathcal{A}}^i$) denote the multi modal assignment problem defined by the two steps above. Due to Proposition 1 we have the following.

Corollary 2. *The minimal cost multi modal assignment in MMAP($\hat{\mathcal{A}}^i$) can be found by solving a classical assignment problem, denoted AP($\hat{\mathcal{A}}^i$), using the minimal cost of each cell in MMAP($\hat{\mathcal{A}}^i$).*

Due to Corollary 2, we can use an algorithm for ranking classic linear assignments with the slightly more general branching technique described above. An efficient algorithm for ranking classic assignments is given by Pedersen et al. [19]. The algorithm uses a reoptimization solution technique such that the minimal cost assignments for the subsets can be found easily (see [19] for more details). Since the general branching technique described above does not create more subsets than the classic branching technique, the overall complexity for ranking the K best multi modal assignments is the same.

Corollary 3. *The complexity for finding the K best assignments is $\mathcal{O}(Kn^3)$.*

Actually, in some cases the minimal cost assignment for subset $\hat{\mathcal{A}}^i$ can be found without solving an AP. Given subset $\hat{\mathcal{A}}$, assume without loss of generality that each cell (i,j) in MMAP($\hat{\mathcal{A}}$) contains L_{ij} entries $c_{ij1} \leq \ldots \leq c_{ijL_{ij}}$ (not set to infinity). Moreover, let $\hat{u}$ and $\hat{v}$ denote the dual row and column variables of the optimal assignment (14) found by solving AP($\hat{\mathcal{A}}$). Hence the corresponding reduced cost for each cell (i,j) becomes $\hat{c}_{ij} = c_{ij1} - \hat{u}_i - \hat{v}_j$. If we disregard cell (i,j), the minimum reduced costs in row i and column j are

$$R_i = \min_t\{\hat{c}_{it} \mid t \neq j\} \text{ and } C_j = \min_t\{\hat{c}_{tj} \mid t \neq i\}.$$

Note that $R_i, C_j \geq 0$, $\forall i,j$, due to optimality of $\hat{u}$ and $\hat{v}$. Now, consider subset $\hat{\mathcal{A}}^i$. In MMAP($\hat{\mathcal{A}}^i$) we set c_{ij_i1} to infinity. If $L_{ij_i} > 1$, we replace c_{ij_i1} with c_{ij_i2} in AP($\hat{\mathcal{A}}^i$). That is, AP($\hat{\mathcal{A}}^i$) uses the same costs as AP($\hat{\mathcal{A}}$) except in cell (i,j_i) where c_{ij_i2} is used. We have the following proposition to enhance the performance of our procedure.

Proposition 7. *Assume $L_{ij_i} > 1$ and $R_i + C_{j_i} \geq c_{ij_i2} - c_{ij_i1}$. Then an minimal cost assignment for subset $\hat{\mathcal{A}}^i$ is*

$$\hat{a}^i = (\hat{a} \setminus \{(i,j_i,1)\}) \cup \{(i,j_i,2)\}$$

Proof. Define

$$\Delta_{ij_i} = c_{ij_i2} - c_{ij_i1} \geq 0$$

The assignment $\hat{a}$ of AP($\hat{\mathcal{A}}$) is primal feasible and satisfies the complementary slackness conditions $\hat{x}_{ij1}\hat{c}_{ij} = 0$. Consider AP($\hat{\mathcal{A}}^i$) with dual row and column variables $\bar{u}$ and $\bar{v}$. If $\Delta_{ij_i} \leq R_i$, set $\bar{u}_i = \hat{u}_i + \Delta_{ij_i}$ and keep the remaining dual values unchanged. Then $\bar{c}_{it} \geq 0$, $\forall t \neq j_i$ and

$$\bar{c}_{ij_i} = c_{ij_i2} - (\hat{u}_i + \Delta_{ij_i}) - \hat{v}_{j_i} = \hat{c}_{ij_i} = 0$$

Hence the assignment $\hat{a}^i$ of AP($\hat{\mathcal{A}}^i$) is primal feasible and satisfies the complementary slackness conditions, i.e. $\hat{a}^i$ is an optimal solution.

If $\Delta_{ij_i} > R_i$, we set $\bar{u}_i = \hat{u}_i + R_i$ and $\bar{v}_{j_i} = \hat{v}_{j_i} + \Delta_{ij_i} - R_i$ and keep the remaining dual values unchanged. Hence $\bar{c}_{it} \geq 0$ and

$$\bar{c}_{ij_i} = c_{ij_i2} - (\hat{u}_i + R_i) - (\hat{v}_{j_i} + \Delta_{ij_i} - R_i) = \hat{c}_{ij_i} = 0$$

Again, assignment $\hat{a}^i$ is optimal. □

Using Proposition 7, we do not have to solve AP($\hat{\mathcal{A}}^i$) if $R_i + C_{j_i} \geq c_{ij_i2} - c_{ij_i1}$. The minimal cost assignment $\hat{a}^i$ is simply obtained by assigning the rows to the same columns as in assignment $\hat{a}$ and, in cell (i,j_i), by using entry 2 instead of entry 1.

5 Computational results

In this section, we report the computational experience on BiMMAP test instances. Moreover, since BiMMAP is an extension of BiAP, we also report some results on test instances for BiAP. All tests were performed on an Intel Xeon 2.67 GHz computer with 6 GB RAM using a Red Hat Enterprise Linux version 4.0 operating system.

5.1 Implementation details

The algorithms have been implemented in C++ and compiled with the GNU C++ compiler using optimize option `-O`.

The cost matrix of BiMMAP (see Figure 1) is stored using a two-dimensional array of Cell objects. Each Cell object contains an array holding the cost entries and an ordered array holding the parametric costs of the entries for a specific λ.

In phase one, for a given search direction specified by λ, we update the parametric costs and order them in nondecreasing order. Due to Proposition 1, we consider the smallest entry in each cell and solve the resulting AP using the implementation given by Jonker and Volgenant [11]. Furthermore, we take advantage of Proposition 2 or Proposition 4 (if $\varepsilon > 0$) whenever possible.

In phase two, we, as in phase one, update the parametric costs and order them in nondecreasing order for a given search direction specified by λ. Next, we use the K best multi modal assignment procedure described in Section 4 to search a triangle, using the upper bounds given in Proposition 3 or Proposition 5 (if $\varepsilon > 0$).

The K best multi modal assignment procedure was implemented using the re-optimization algorithm in Pedersen et al. [19] for ranking classic assignments, with the slightly more general branching technique given in Section 4. In particular, note that, when considering a subset where we remove an entry in the ordered array of parametric costs, the new entry with minimal cost is the next cost in the array. That is, we just have to increase a local pointer by one to find the new minimal cost. See Pedersen et al. [19] for more details on the ranking implementation.

All nondominated points found by the ranking procedure are stored in a single linked list available in both phase one and two.

5.2 BiMMAP test instances

The bicriterion multi modal assignment problem has, to the best of our knowledge, not previously been studied in literature, and hence no available test instances exist for this problem. To facilitate a comprehensive computational study of our BiMMAP algorithm, we build a problem generator, *APGen*, for this problem class.[3] As a side-effect our generator can be used to generate a variety of BiAP instances. In the following we give a brief description of the generator, and we refer readers requiring more information on this topic, to the full documentation paper [17]. A BiMMAP instance is generated specifying a number of parameters:

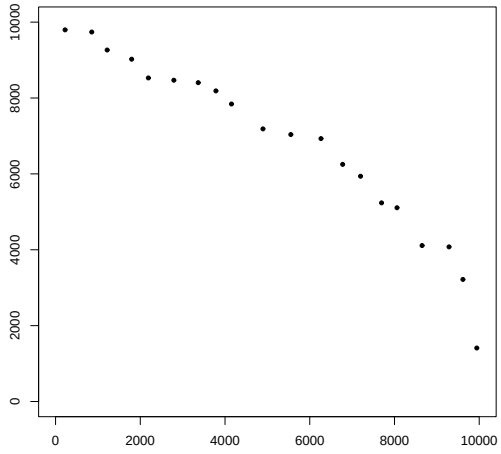
n – the size of the problem.

$maxEnt$ – the maximal number of entries in each assignment cell.

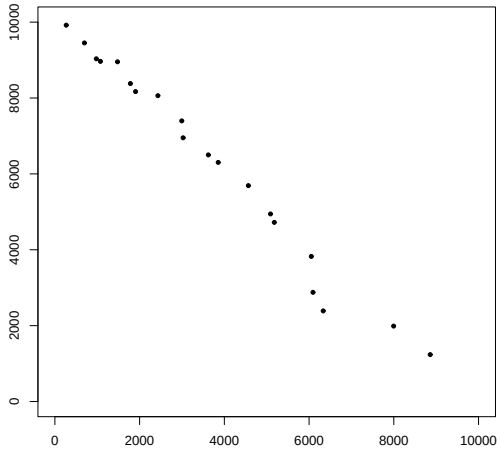
$minEnt$ – the minimal number of entries in each assignment cell (default 1).

$maxCost$ – the maximal cost value for both objectives (minimum cost value is 0).

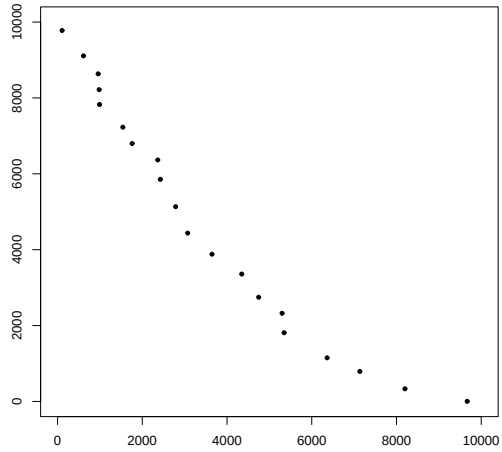
[3]The problem generator and the test instances used in this paper are downloadable from the following webpage `http://www.research.relund.dk/`.



(a) $shape = -60$



(b) $shape = 0$



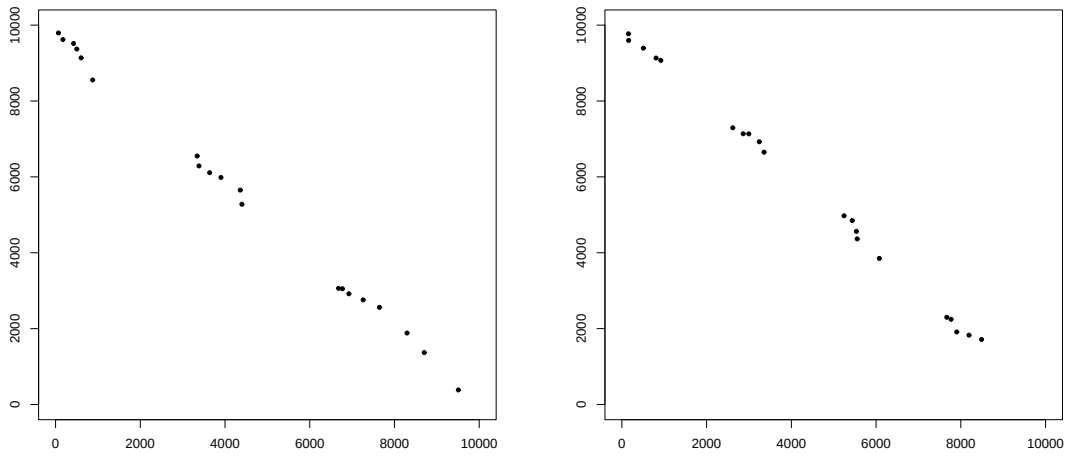
(c) $shape = 60$

Figure 8: Cell entries for $method = 1$.

method – a choice between three different ways of generating cell entries.

shape – for a given method, the shape parameter describes the shape of the entries in a given cell.

Obviously, for a given cell, no entries are allowed to be dominated by other entries in that cell, since this would correspond to a dominated solution. The number of entries in a cell is chosen randomly in the *entry range* $\{minEnt, \ldots, maxEnt\}$. To describe best the six different versions of *method* and *shape* used for generating BiMMAP instances, we have displayed all two-dimensional cost vectors for a given cell having 20 entries in Figures 8 to 10.



(a) *shape* = 3 (b) *shape* = 4

Figure 9: Cell entries for *method* = 2.

As can be seen in Figure 8, with method 1 the shape parameter describes the curve of the function along which the entries are generated. A negative shape corresponds to generating the entries along a concave-like function, using shape 0 generates entries fluctuating along a straight line, and finally, a positive shape means generating entries along a convex-like function. Therefore, using a negative (positive) shape parameter tends to generate many unsupported (supported) entries in the given cell. We shall see, that this has a strong influence on the difficulty of the considered problem and hence the computational performance of our algorithm.

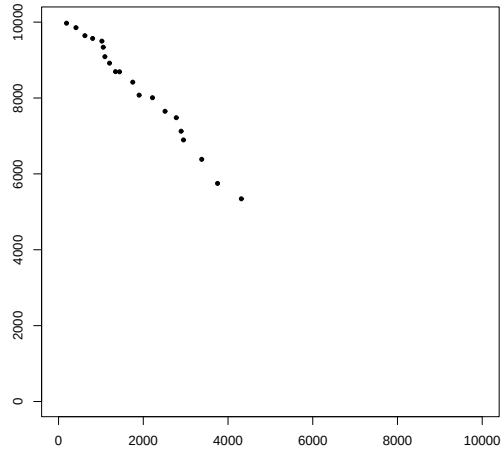
For method 2, the entries in a given cell are generated in a number of groups specified by the shape parameter (see Figure 9). Note, to use method 2, the parameter *minEnt* must be chosen sufficiently large, since at least 2 points have to be in each group.

Finally, for method 3, the shape parameter has the same meaning as for method 1. However, the entries fall either in the upper left corner of the cost-space or the lower right corner in consecutive cells. This can be seen in Figure 10, where we display the entries in the four consecutive assignment cells $(1,1),\ldots,(1,4)$.

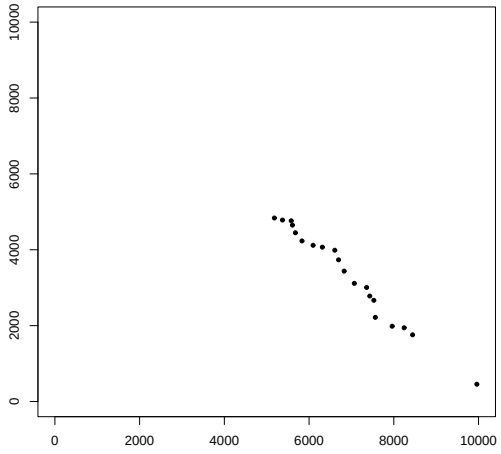
To provide a broad class of test instances and facilitate statistical analysis, we generated 100 instances of each of the following 80 possible configurations.

- $n \in \{4,6,8,10\}$.
- Cost ranges : $\{0,\ldots,500\}$ and $\{0,\ldots,10000\}$.
- Entry ranges : $\{2,\ldots,8\}$ (not for method 2) and $\{10,\ldots,30\}$.
- $(method, shape) \in \{(1,-60),(1,0),(1,60),(2,3),(2,4),(3,0)\}$.

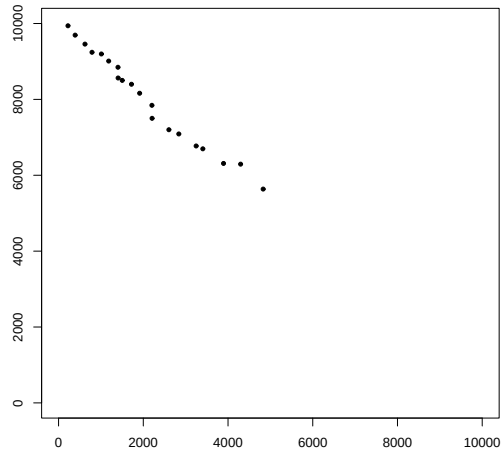
The two different ranges of number of entries are chosen to reflect a situation close to BiAP (few entries) and a situation very far away from BiAP (many entries),



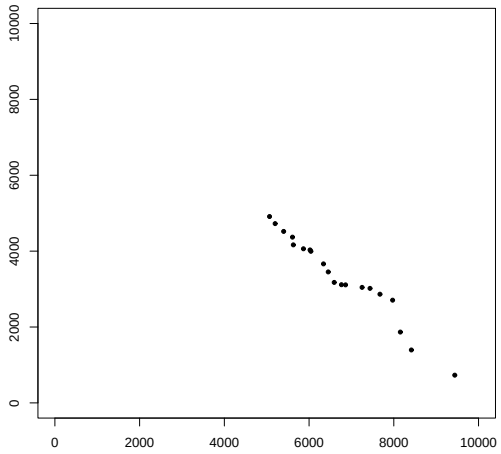
(a) cell $(1, 1)$



(b) cell $(1, 2)$



(c) cell $(1, 3)$



(d) cell $(1, 4)$

Figure 10: Cell entries for cells $(1, 1)$ to $(1, 4)$ for $method = 3$ and $shape = 0$.

respectively. Note that the number of possible assignments increases exponentially with the number of entries in each cell. For a problem with $n = 10$ and entry range $\{10, \ldots, 30\}$, the total number of assignments ranges between $10! \cdot 10^{10} \approx 3.6 \cdot 10^{16}$ in the best case and $10! \cdot 10^{30} \approx 3.6 \cdot 10^{36}$ in the worst case. In comparison, the BiAP with size 10 has only $10! \approx 3.6 \cdot 10^6$ feasible assignments.

5.3 BiMMAP test results

Giving the results of the extensive amount of tests, we first display the logarithm of the CPU time (in seconds) averaged over the 100 instances against problem size n for

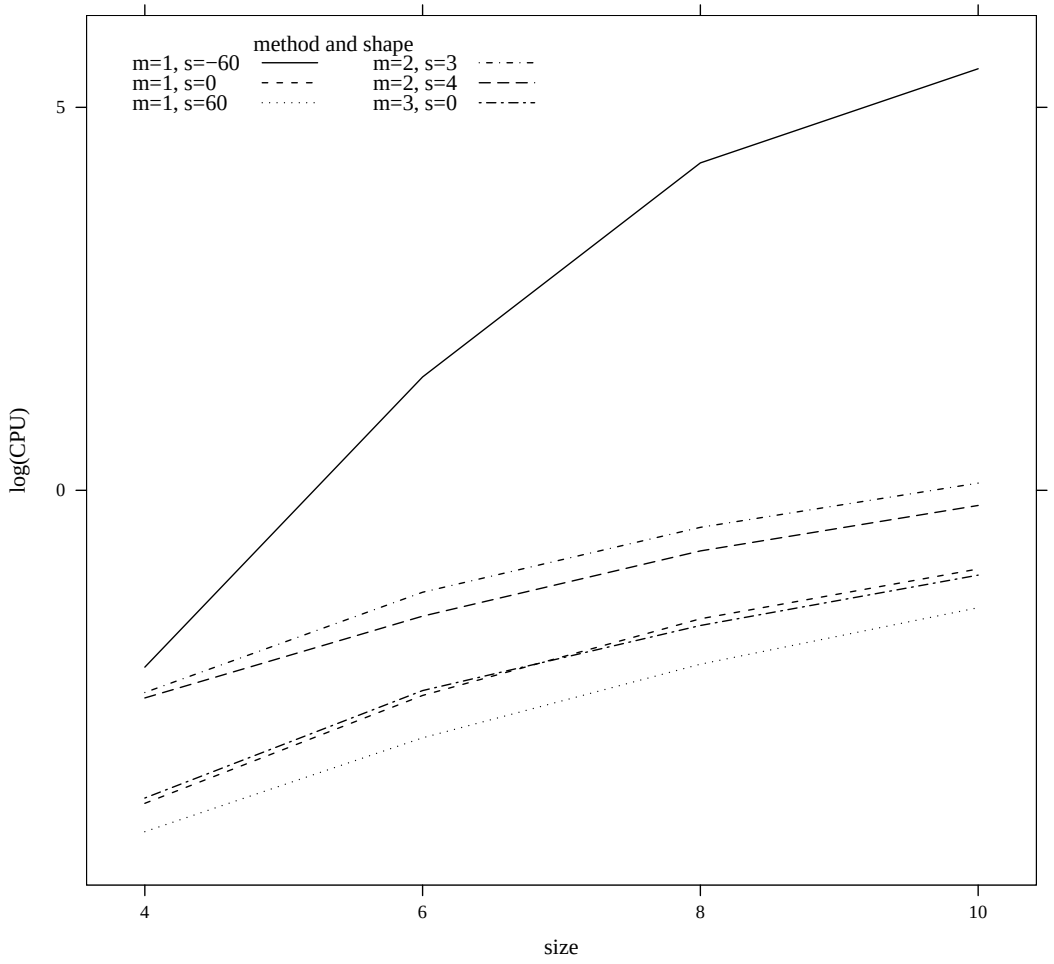
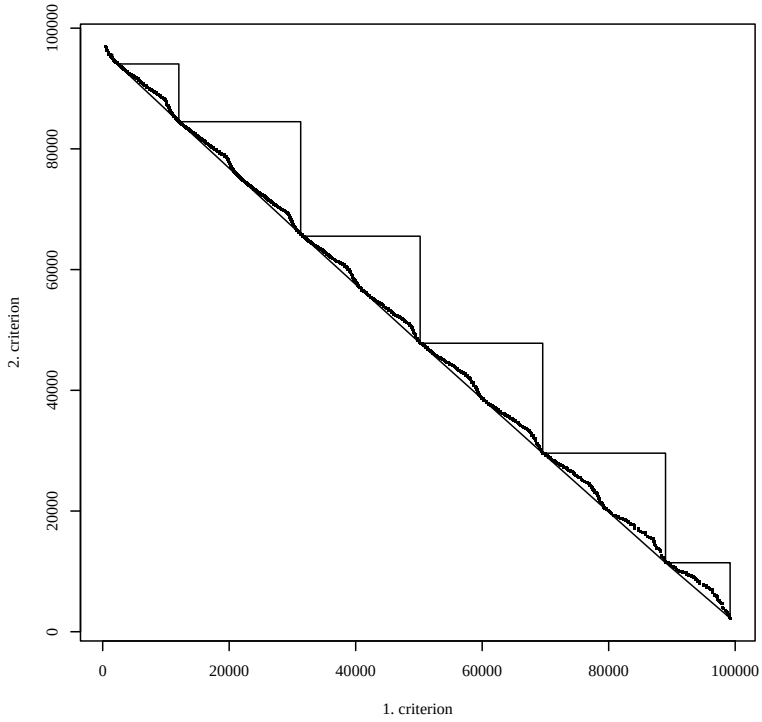


Figure 11: Logarithm of average CPU against n (cost range $\{0, \ldots, 10000\}$ and entry range $\{10, \ldots, 30\}$).

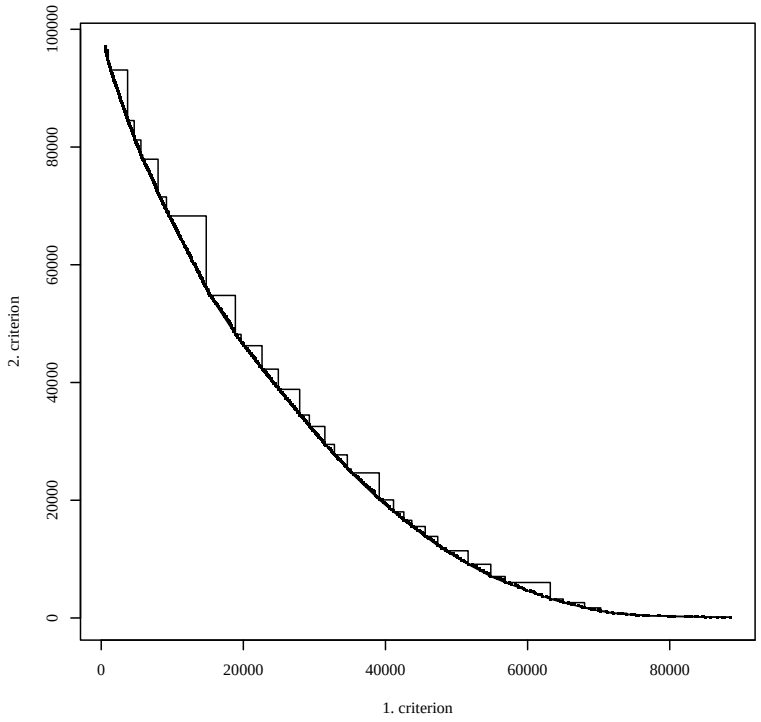
the highest possible cost range and the highest possible entry range (Figure 11). It can be seen, that, for none of the six different classes, the running time is increasing exponentially with problem size. Also notice that the most difficult class is by far using method 1 with shape -60, whereas the easiest class is method 1 and shape 60.

To yield a possible explanation of the difference in difficulty of these two problem classes, we direct the attention of the reader to Figure 12 where the nondominated points in the criterion space have been plotted for two test instances using $shape = -60$ and $shape = 60$, respectively. Triangles are drawn between consecutive supported extreme nondominated points.

For the test instance with $shape = -60$, only a limited number of supported extreme nondominated criterion points exist. Notice that these extreme points are far from each other resulting in large triangles to search in the second phase. More important, all the extreme supported nondominated points are almost on a straight line. Therefore, the search directions for the triangles are more or less the same. As a



(a) $shape = -60$



(b) $shape = 60$

Figure 12: Nondominated points ($method = 1$, $n = 10$, entry range $\{10, \ldots, 30\}$ and cost range $\{0, \ldots, 10000\}$).

result, the ranking procedure initiated in the first large triangle has to generate many points before reaching the upper bound of the triangle making this single triangle search extremely time consuming. Remember though, that nondominated points generated which are outside the triangle currently searched, are stored. This may enable us to finish searching other triangles faster and hence enhance computational performance.

In contrast, the test instance with $shape = 60$ has more extreme nondominated points in the criterion space, resulting in small triangles to search. Moreover, the search directions are more diverse and hence fewer points have to be generated when searching a triangle.

Considering other instances the above relationships proved to have general validity. The larger a triangle is, the longer the search in this triangle continues in phase two. Therefore, with many extreme points at termination of phase one, only small triangles need to be searched, resulting in a lower overall running time compared to an instance with a few extreme points and larger triangles. Moreover, test instances where the search directions for the triangles are more or less the same are harder to solve.

Comparing running times of phase one and two, phase two can be seen to be the major time consumer. On average, phase two uses 98 per cent of the total CPU time in the 8000 exactly solved instances.

Since method 1 shape -60 has established itself as the most difficult problem class, we focus on this instance only from here on.

In Figure 13, we show the logarithm of average CPU time against the problem size for the four different configurations of entry range and cost range. It can be seen, that the most difficult case is the one with the most entries and highest cost range. The most significant factor is the entry range, obviously resulting from the increased number of feasible solutions. Also, for a given number of entries, the most difficult case arises with the highest possible cost range. In this respect, the BiMMAP follows the classical single criterion assignment problem since this problem is known to be easiest solvable with relatively small costs [11].

From here on, we focus on test instances using $method = 1$, $shape = -60$ cost range $\{0, \ldots, 10000\}$ and entry range $\{10, \ldots, 30\}$ only. In Table 1, we give the numerical results for the exact solution of these instances. In the six columns are depicted the size of the problem, average CPU time (seconds), maximal CPU time, average number of supported nondominated points, average number of unsupported nondominated points and average of the total number of nondominated points, respectively. Obviously, all columns are increasing in size. However, it is interesting to note the relatively high number of unsupported nondominated criterion points.

Now we describe the results for finding an approximation of the nondominated set. Two small values 0.01 and 0.05 of ε are chosen to ensure that a sufficiently accurate approximation is found. We also include the results for the exact solved instances ($\varepsilon = 0$). In Figure 14, we display the logarithm of average CPU time against size for all three ε values.

Finding an approximation can be seen to have a strong influence on the running time of the algorithm. Even for these small ε values (and hence good approximations) there are significant savings in computation time. We note (not displayed)

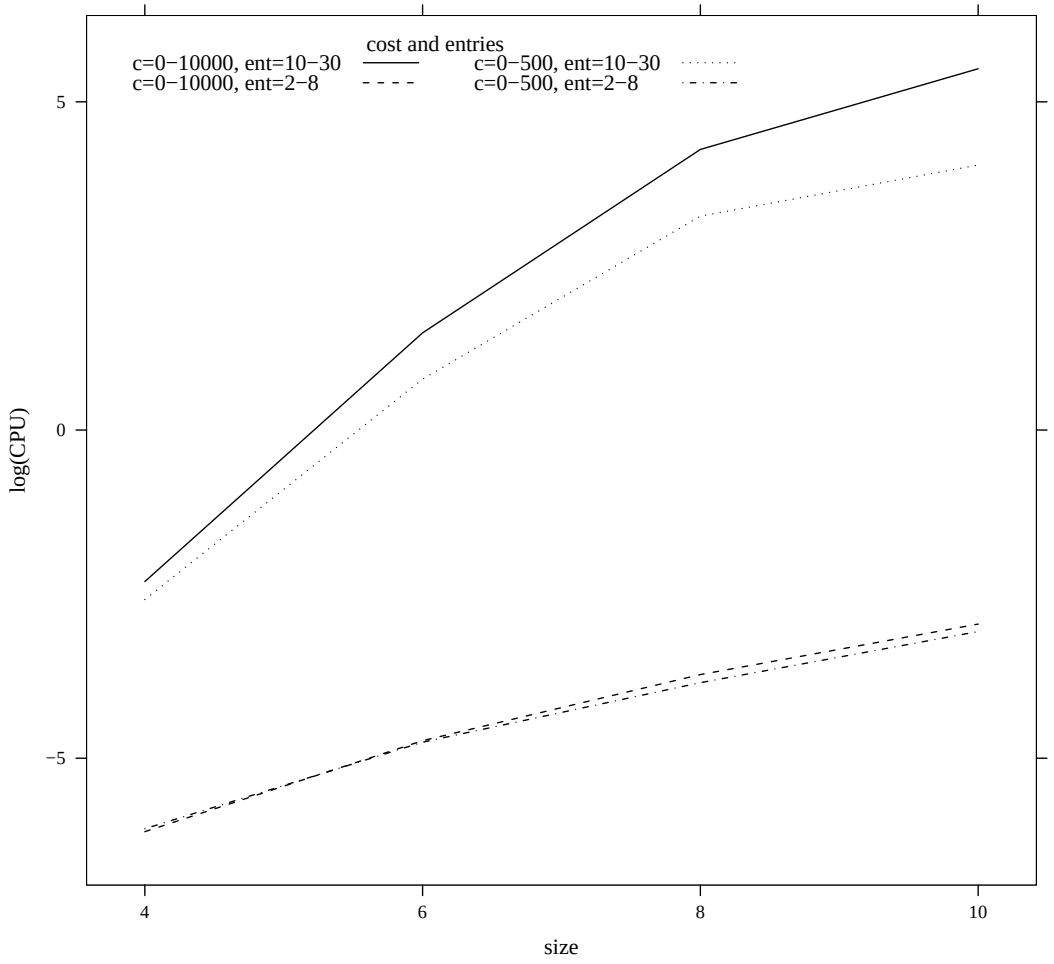


Figure 13: Logarithm of average CPU against n ($method = 1$ and $shape = -60$).

size	ave CPU	max CPU	ave SND	ave USND	ave ND
4	0.10	0.34	6.95	210.04	216.99
6	4.39	33.21	9.87	437.30	447.17
8	71.84	559.28	13.07	744.34	757.41
10	245.63	2967.09	16.23	1143.72	1159.95

Table 1: Exact results ($method = 1$, $shape = -60$, entry range $\{10, ..., 30\}$ and cost range $\{0, ..., 10000\}$).

that the number of identified nondominated points obviously decreases with increasing ε, as some extreme supported nondominated points may not be considered in the first phase, and in the second phase, fewer alternatives for each triangle are ranked.

In Figure 15, we graph the empirical distribution functions of CPU time for the 100 test instances for problem size 10. This clearly shows that the majority of

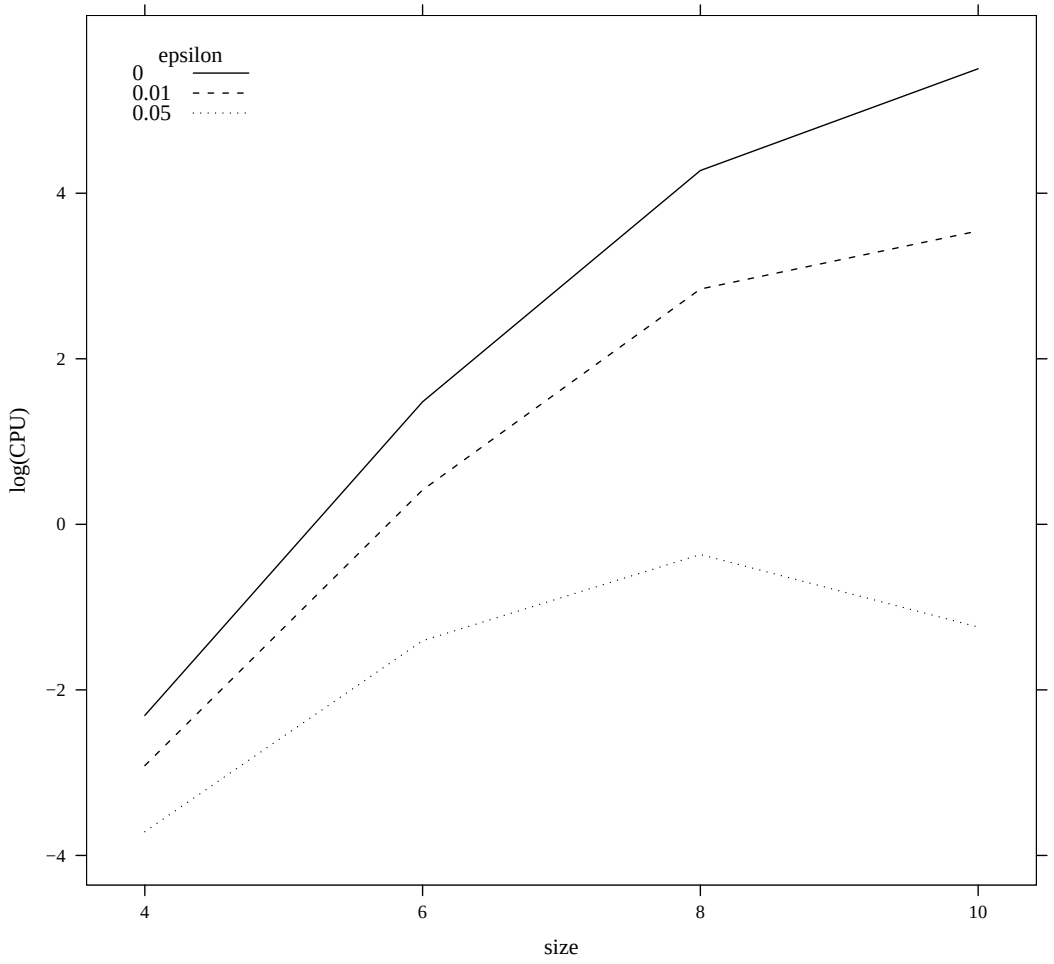


Figure 14: Logarithm of average CPU against n ($method = 1$ and $shape = -60$).

	$\varepsilon = 0$			$\varepsilon = 0.01$			$\varepsilon = 0.05$		
size	ave.	90%	max	ave.	90%	max	ave.	90%	max
4	0.10	0.18	0.34	0.05	0.10	0.17	0.02	0.05	0.09
6	4.39	9.19	33.21	1.52	3.01	12.25	0.25	0.52	2.01
8	71.84	163.16	559.28	17.16	41.71	172.38	0.69	1.60	6.01
10	245.63	589.52	2967.09	34.61	85.72	420.15	0.29	0.66	2.82

Table 2: CPU times for $\varepsilon = 0$, 0.01 or 0.05 ($method = 1$, $shape = -60$, entry range $\{10, ..., 30\}$ and cost range $\{0, ..., 10000\}$.

problems are solved fast, while only a few difficult instances are solved relatively slowly. The numerical results are summarized in Table 2, giving for each ε the average CPU time (seconds), the 90 per cent fractile of CPU time and the maximum CPU time.

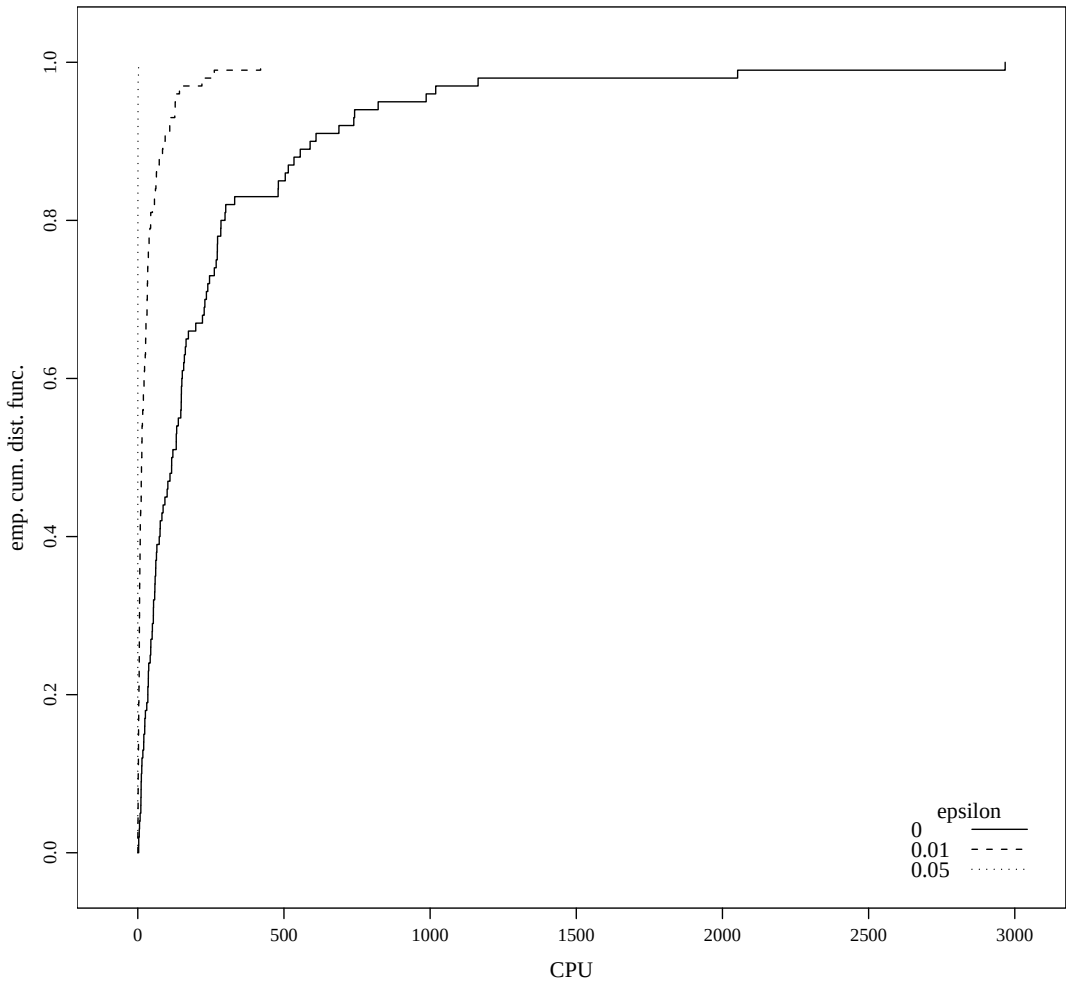


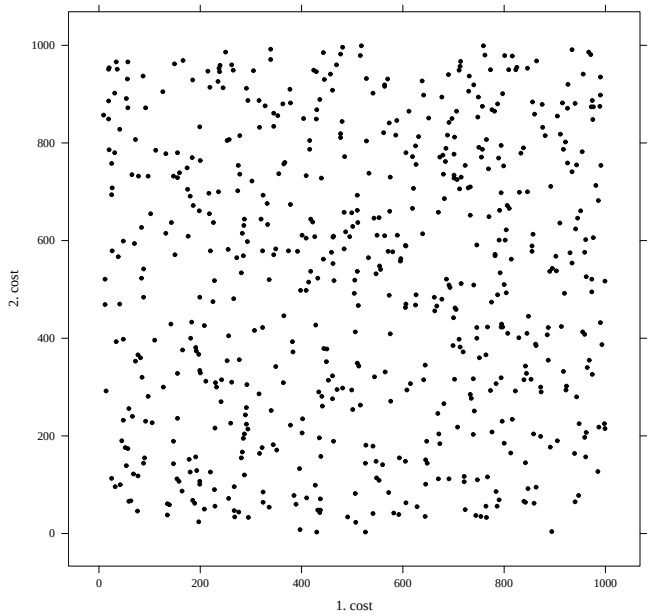
Figure 15: Empirical distribution of CPU time for different ε values.

5.4 Results for BiAP

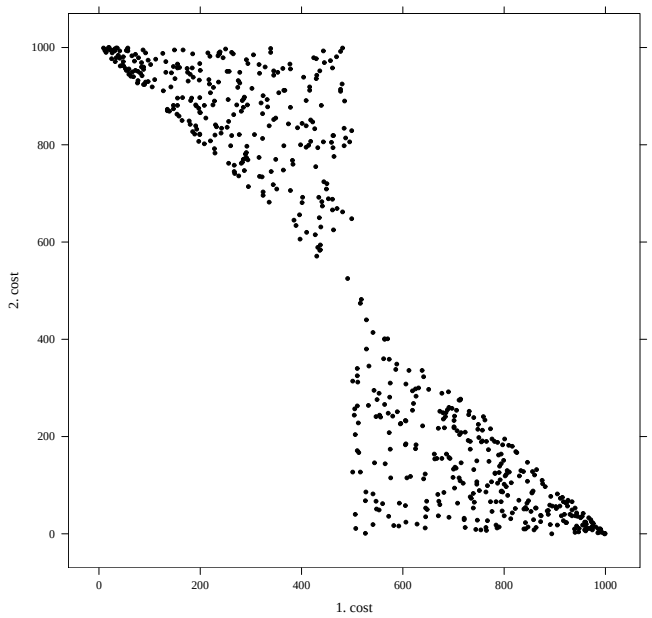
Since BiMMAP is an extension of BiAP, we found it natural to test the performance of our current implementation on this problem class. To yield consistency in literature, we obtained the test instances used in [23] which are BiAP instances of size $\{5, 10, \ldots, 50\}$. Also previously used in literature are AP instances of size $\{60, 70, \ldots, 100\}$ found in [8].[4] These instances have recently been solved by an exact method in [20] and by a heuristic in [9] acknowledging the current interest in this field. For all the problem classes only one instance is solved, and hence limited statistics can be performed for those data sets. For all instances, costs are chosen randomly in the rather narrow interval $\{0, \ldots, 19\}$.

To provide our reader with statistics based on a broader class of instances, we generated 100 instances of each of the following sizes $\{5, 10, \ldots, 100\}$ with costs randomly chosen in $\{0, \ldots, 1000\}$. This wide interval leaves room for identifying a large number of large triangles to search in phase two, and hence adds to the

[4] `http://www.univ-valenciennes.fr/ROAD/MCDM/ListMOAP.html`



(a) random.



(b) negatively correlated.

Figure 16: Cost generation for BiAP test instances.

difficulty of the problem. Also, to investigate the effect of negatively correlated costs, we generated 100 instances of each of the sizes $\{5, 10, \ldots, 100\}$, again with costs in the interval $\{0, \ldots, 1000\}$. The difference in costs generated randomly and negatively correlated is shown in Figure 16. We shall see that negatively correlated cost vectors have a strong influence on the difficulty of the considered problem, and hence the running time of the algorithm. In general bicriterion problems with negatively correlated costs are harder to solve, see e.g. [4].

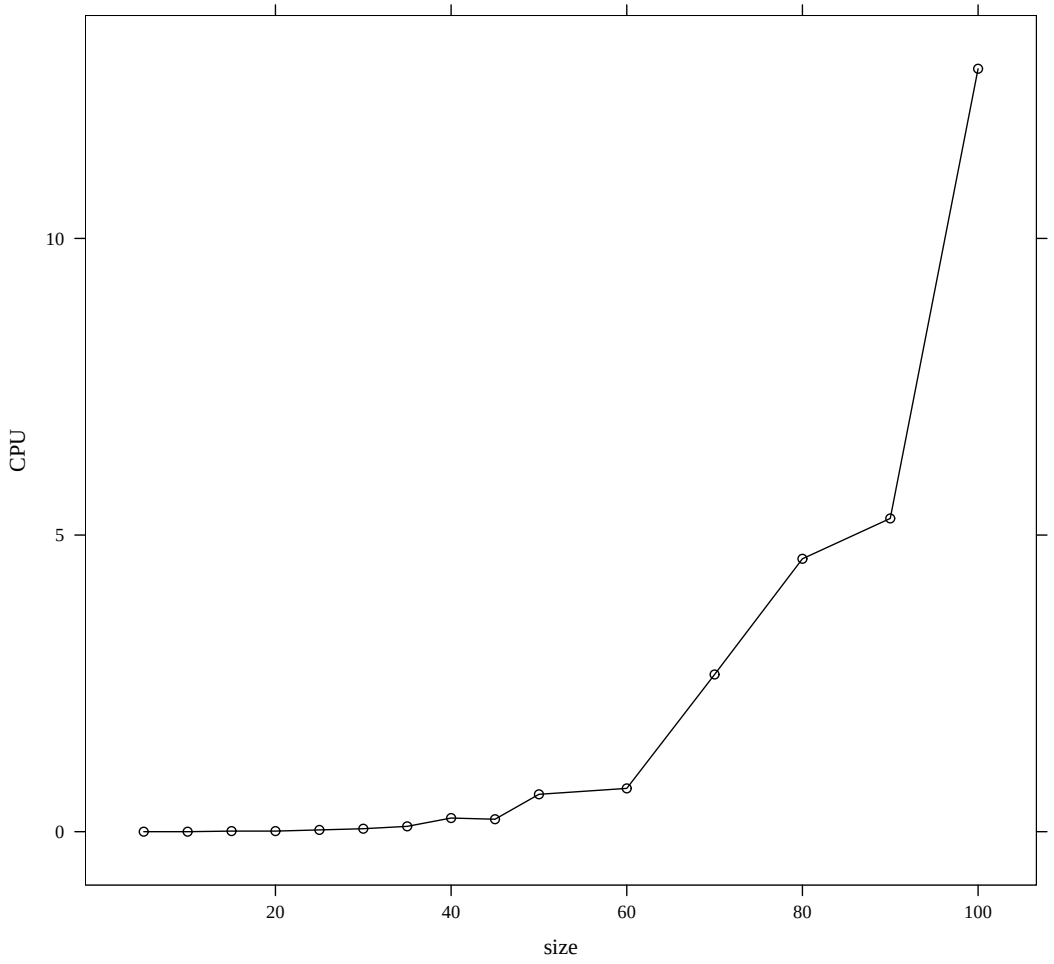


Figure 17: CPU time against n for the instances found in [8] and [23].

In Figure 17, we graph CPU time against size for the instances previously found in literature. We note that the number of nondominated points we found corresponds exactly to the number of efficient solutions found by CPLEX in [8], making this set a minimal complete set of efficient solutions. In [8], the results were already concluded to be questioning the validity of the results from [23], and this is substantiated by our results. A comparison of our CPU time with the CPU time of the exact algorithms reported upon in [8, 20, 23] must necessarily include a discussion on the efficiency of the different computers used. Applying *Linpack Benchmark-Peaks* from Netlib [15] to reflect the relative performance of the computers, we can see that our algorithm outperforms the exact methods previously proposed in literature. Furthermore, the running time of our exact method is even competitive to the CPU time for the heuristics proposed in [8, 9].

Figure 18, shows average CPU time against size for the negatively correlated and random BiAP test instances. For the negatively correlated instances, we were only capable of solving instances of problem size up to 40 within a reasonable amount of time. This shows the complex nature of such instances, as is also previously seen for

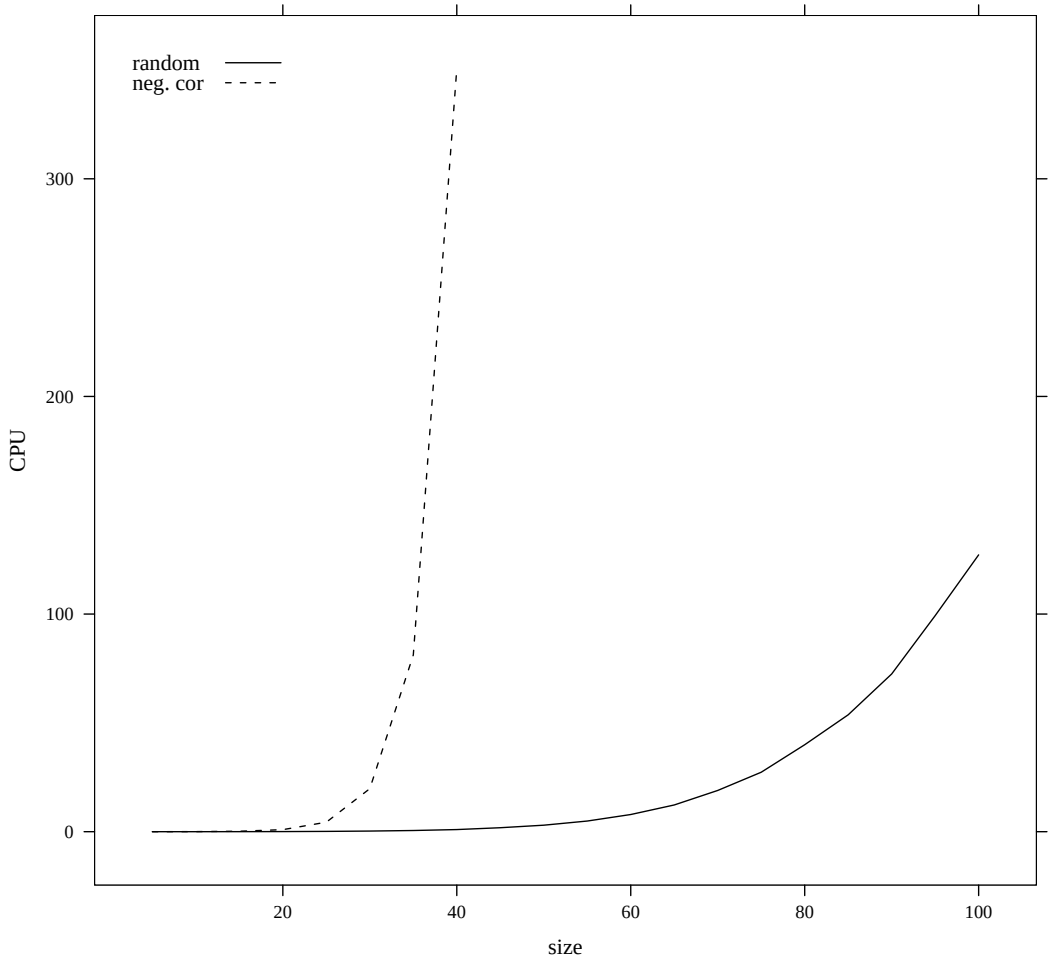


Figure 18: Average CPU time against n for our BiAP instances.

other bicriterion problems. The increased difficulty follows mainly because we have more (see Table 3) and larger triangles (not shown). Moreover, note that the number of nondominated points is much higher. For the size 40 problem reported upon in [23], a total of 127 nondominated points was found, 73 of which were unsupported. In Table 3, we see that the average number of nondominated points and the average number of unsupported nondominated points for the size 40 negatively correlated instances (random instances) are 2402.04 (333.24) and 2346.02 (296.55), respectively.

Having CPU times no larger than 182 seconds for the random instances and 2635 seconds for the negatively correlated instances, our algorithm proves capable of solving BiAP problems rather efficiently.

Let us conclude this section by commenting on some numerical statistics obtained for the BiMMAP instances and for the different BiAP problems. To collect outputs for making such statistics, we intentionally neglected to focus on CPU time by disabling the IP strengthening of the upper bound used when searching a triangle. This makes the ranking process run much longer resulting in a significant increase in memory usage.

	neg. corr. data			random data		
size	ave SND	ave USND	ave ND	ave SND	ave USND	ave ND
5	5.56	14.86	20.42	3.67	2.44	6.11
10	13.23	115.44	128.67	7.80	16.58	24.38
15	20.12	294.78	314.90	12.23	40.83	53.06
20	27.89	539.18	567.07	17.56	72.43	89.99
25	34.95	877.59	912.54	22.68	118.64	141.32
30	41.83	1298.96	1340.79	27.17	167.41	194.58
35	48.63	1773.25	1821.88	32.65	230.82	263.47
40	56.02	2346.02	2402.04	36.69	296.55	333.24

Table 3: Exact results for our BiAP instances.

Calculating the number of times the IP upper bound would succeed in terminating the search of a triangle when the bound without IP strengthening would fail and dividing this by the total number of iterations, the average ratio for all 8000 exactly solved BiMMAP instances is 11.47 per cent. This corresponds to saying that, in more than one out of 10 cases the search in a given triangle would be terminated when using the integer based upper bound, whereas the upper bound without integrality would still be higher than the lower bound. The same ratio for our negatively correlated BiAP instances is 5.80 per cent, for our random instances 7.67 per cent and finally, for the random BiAP instances found in literature, 79.55 per cent. Obviously, since only a limited amount of instances have previously been reported upon in literature, this last number must be interpreted with caution. However, it seems like the IP strengthening proves to be valuable for primarily instances having a narrow cost range. This yields another explanation for our instances being significantly harder than the instances previously seen in literature.

It is also interesting to consider the ratio between the number of efficient solutions and the number of nondominated points. For BiMMAP instances, this ratio has an average of 1.01 and a maximum of 1.25 found for a size 8 instance using the lowest possible cost range $\{0, \ldots, 500\}$ and highest possible entry range $\{10, \ldots, 30\}$. This fits nicely with our intuition that a narrow cost range leads to many alternative efficient solutions. Moreover, the high number of entries also yields a higher total number of feasible solutions. For the 15 random generated BiAP instances from [8, 23], the average of this ratio is 2.11, and for both our random instances and our negatively correlated instances, this ratio is very close to 1. The main factor here is again the cost range. A narrow cost range gives a higher number of alternative efficient solutions corresponding to the same nondominated point.

6 Conclusion

In this paper we have presented a two-phase method for solving a new bicriterion generalization of the classical linear assignment problem. The algorithm uti-

lizes a ranking scheme relying on reoptimization, which was previously shown to be very efficient. Exploiting the integral nature of the criterion vectors allowed us to strengthen an upper bound previously stated in literature. Our algorithm identifies either the exact set of all nondominated points or an approximation of these with predetermined approximation quality.

A large library of test instances for both BiMMAP and BiAP was provided by a new problem generator, and a diversity of numerical results was presented. Below we summarize the main results.

For BiMMAP the excessive group of test problems allowed us to identify the instances with cell entries generated along a concave-like function (method 1 and shape -60) to be significantly harder than any other problem class. This mainly follows since the many unsupported cost vectors in each cell generate few extreme supported points located along an almost straight line in criterion space. Therefore, the search directions for the individual triangles become close to identical, resulting in long running times for the first large triangle investigated. Since phase two was established as the main time consumer, this explains why such instances are very difficult. In contrast, the instances with many supported cell entries (method 1 and shape 60) generate more supported extreme criterion vectors located along a convex-like function. Therefore, the triangles to search in the second phase are smaller and have more diverse search directions, making these instances easier.

Considering two different ranges of costs and of number of entries in each assignment cell, it was pointed out that high costs and many cell entries result in the most difficult instances. This was concluded to fit nicely with results for the single criterion AP and with the fact that the total number of feasible solutions grows rapidly with the number of cell entries.

For BiAP, the main knowledge gained by our numerical experiments is that instances with negatively correlated cost vectors by far exceed random BiAP instances in difficulty. This mainly follows due to the higher number of large triangles to search in the time-consuming second phase.

Our algorithm was concluded to perform very well even on BiMMAP instances of high size, and was seen to outperform existing methods for BiAP. The integral strengthening of the upper bound proved to be contributing significantly to the efficiency of our algorithm. Finding an approximation of the nondominated set was seen to be a valuable tool to reduce the computation time significantly. The results reported here show the approximative algorithm to be a serious rival to heuristical methods. Note, applying an approximative scheme instead of a heuristical method allows us to control the quality of the reported set of criterion points.

A natural extension of BiMMAP is the bicriterion multi modal transportation problem which interests the authors of this paper. Also, under current investigation is a bicriterion version of the directed Chinese Postman Problem (BiDCPP). Restricting the deviation in in- and out-degree for all nodes in the original postman graph to be no larger than one, BiDCPP can be seen to yield an instance of BiMMAP as a subproblem.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] Y.P. Aneja and K.P.K. Nair. Bicriteria transportation problem. *Management Science*, 25(1):73–78, 1979.

[3] J. Cohen. *Multiobjective Programming and Planning*. Academic Press, New York, 1978.

[4] C. Gomes da Silva, J. Clímaco, and J. Figueira. A scatter search method for bicriteria $\{0,1\}$-knapsack problems. *European Journal of Operational Research*, 169(2):373–391, 2006.

[5] M. Dell'Amico and S. Martello. Linear assignment. In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 355–371. Wiley, Chichester, 1997.

[6] M. Dell'Amico and P. Toth. Algorithms and codes for dense assignment problems: the state of the art. *Discrete Appl. Math.*, 100(1-2):17–48, 2000. doi: 10.1016/S0166-218X(99)00172-9.

[7] M. Ehrgott. *Multicriteria optimization*, volume 491 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, 2000.

[8] X. Gandibleux, H. Morita, and N. Katoh. Use of genetic heritage for solving the assignment problem with two objectives. *Lecture Notes in Computer Science*, 2632:43–57, 2003.

[9] X. Gandibleux, H. Morita, and N. Katoh. A population-based metaheuristic for solving assignment problems with two objectives. To appear in Journal of Mathematical Modelling and Algorithms, Revised 2005.

[10] P. Hansen. Bicriterion path problems. In *Multiple Criteria Decision Making, Theory and Application*, number 177 in Lecture Notes in Economics and Mathematical Systems, pages 109–127. Springer-Verlag, Berlin, 1979.

[11] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

[12] H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quart.*, 2:83–97, 1955.

[13] H.W. Kuhn. Variants of the hungarian method for the assignment problem. *Naval Research Logistics Quart.*, 3:253–258, 1956.

[14] K.G Murty. An algorithm for ranking all the assignments in order of increasing cost. *Operations Research*, 16:682–687, 1968.

[15] Netlib. The performance database server. URL `http://netlib.org/`.

[16] L.R. Nielsen, K.A. Andersen, and D. Pretolani. Bicriterion shortest hyperpaths in random time-dependent networks. *IMA Journal of Management Mathematics*, 14(3):271–303, 2003.

[17] L.R. Nielsen and C.R. Pedersen. APGen - an assignment problem generator. URL `http://www.research.relund.dk`. Reference manual.

[18] M. Pascoal, M. Eugénia Captivo, and J. Clímaco. A note on a new variant of Murty's ranking assignments algorithm. *4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(3):243–255, 2003. doi: 10.1007/s10288-003-0021-7.

[19] C.R. Pedersen, L.R. Nielsen, and K.A. Andersen. A note on ranking assignments using reoptimization. Working Paper No. 2005/2, Department of Operations Research, University of Aarhus, 2005.

[20] A. Przybylski, X. Gandibleux, and M. Ehrgott. The biobjective assignment problem. Research Report No 05.07, lina – Laboratoire D'Informatique de Nantes-Atlantique, 2005.

[21] A. Przybylski, X. Gandibleux, and M. Ehrgott. The biobjective integer minimum cost flow problem - incorrectness of Sedeño-Noda and González-Martín's algorithm. Computers and Operations Research, in print, 2005.

[22] P. Serafini. Some considerations about computational complexity for multi objective combinatorial problems. In J. Jahn and W. Krabs, editors, *Recent Advances and Historical Development of Vector Optimization*, volume 294, pages 222–232. Springer, Berlin, 1986.

[23] D. Tuyttens, J. Teghem, Ph. Fortemps, and K. Van Nieuwenhuyze. Performance of the MOSA method for the bicriteria assignment problem. *Journal of Heuristics*, 6(3):295–310, 2000. doi: 10.1023/A:1009670112978.

[24] E.L. Ulungu and J. Teghem. The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of Computing and Decision Sciences*, 20(2):149–165, 1995.

[25] A. Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1):70–79, 1987.