The Bicriterion Multimodal Assignment Problem: Introduction, Analysis, and Experimental Results

Christian Roed Pedersen

Department of Operations Research, University of Aarhus, DK-8000 Aarhus C, Denmark, crpe@tdc.dk.

Lars Relund Nielsen

Department of Genetics and Biotechnology, University of Aarhus, DK-8830 Tjele, Denmark, lars@relund.dk.

Kim Allan Andersen

Department of Business Studies, Aarhus School of Business, University of Aarhus, DK- 8210 Aarhus V, Denmark, kia@asb.dk.

April 10, 2008

Abstract: We consider the bicriterion multimodal 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 are provided.

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

1 Introduction

In general, a description of real-world applications as single-criterion optimization problems is seldom realistic, because they are often by nature imposed with more objectives to be simultaneously optimized. The classical linear assignment problem consists of assigning workers to jobs with total minimum cost. Recently, an important generalization of it has generated considerable interest, namely, assigning workers to jobs with minimal cost and time. This yields the *bicriterion assignment problem* (BiAP). Ulungu and Teghem (1995) 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 (Ulungu and Teghem 1995) was implemented by Tuyttens et al. (2000), showing – with large CPU times – the limitations of this algorithm. Recently, an improvement of this algorithm was given in Przybylski et al. (2008), proposing also a two-phase method applying ranking for BiAP.

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 the literature. Consider 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 minimized simultaneously, namely, *travel time* and *travel or assignment cost*. Because a specialist i has possibly multiple modes of transportation and routes to choose from to reach the destination of job j, several two-dimensional cost vectors exist for each combination of i and j. Therefore, the *bicriterion multimodal assignment problem* (BiMMAP) is an extension of BiAP containing, in each assignment cell, possibly several two-dimensional cost vectors/points.

The objective is to identify either all efficient assignments or all nondominated criterion points for the problem.

Note 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* (Hansen 1979, Gandibleux et al. 2005).

Apart from the above application, BiMMAP also serves as an important subproblem in the *bicriterion directed Chinese postman problem* (BiDCPP). Restricting the deviation of in- and outdegree for all nodes in the original postman graph to be no larger than one, BiDCPP may be solved via a number of bicriterion shortest-path computations followed by the solution of a BiMMAP instance. For more details, refer to Pedersen (2006).

To identify all nondominated criterion points for BiMMAP, we propose a two-phase method. Acknowledging that ranking procedures have been applied with great success for other bicriterion optimization problems (see, e.g., (Nielsen et al. 2003)), we employ ranking of multimodal assignments as a subroutine. The subroutine is an efficient extension of the algorithm for finding the K best assignments by Pedersen et al. (2005a).

A method for finding an ε-approximation (Warburton (1987)) of the set of nondominated points, which enable us to control the quality of the set of criterion points reported, is also presented. It may be the case that an approximation of the set of nondominated points is sufficient when considering a particular application. Moreover, for large problems, it may only be feasible to obtain an approximation 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. Because BiMMAP is a generalization of BiAP, we also report computational results for some BiAP instances previously solved in literature.

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

2 The Bicriterion Multimodal Assignment Problem

In this section, we give the mathematical formulation of (BiMMAP), introduce the relevant terminology, and give a few theoretical results.

2.1 Mathematical Formulation

In BiMMAP, n specialists must be assigned to n jobs such that each specialist performs 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}, $l = 1, \ldots, L_{ij}$. Obviously, BiMMAP is an extension of BiAP where, for each cell (i, j) in the *assignment cost matrix*, we have several two-dimensional cost vectors as illustrated in Figure 1. The objective is to identify either all efficient minimum-cost assignments or all nondominated criterion points for the problem.

Let x_{ijl} be a binary variable with value one, if i is assigned to j using mode choice l, and zero otherwise. Obviously, exactly one i must be assigned to each j using a specific mode choice, which

$i\backslash j$	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.

gives us the following mathematical formulation of BiMMAP:

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

Assume that within each cell (i, j) the costs are mutually nondominated; hence

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

This is no restriction, because a dominated cost vector in a given cell will never be used in an efficient assignment. Moreover, assume that c^1_{ijl} and c^2_{ijl} are integer for all i, j and l.

A feasible solution x to (1) is called a *multimodal assignment*, or for 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 *multimodal 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 $y = (y_1, y_2) = (c^1 x, c^2 x)$ the corresponding objective vector (or criterion vector). Note that if $L_{ij} = 1$ for all i, j, $\mathcal{MMAP}$ reduces to the assignment polytope, $\mathcal{AP}$. Due to the fact that BiMMAP is a generalization of BiAP, it holds true that BiMMAP is $\mathcal{NP}$-complete (Ehrgott 2000, Serafini 1986).

2.2 Methodology

For single-criterion optimization, the concept of optimality is well defined. 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 $y^1 \leq y^2$.

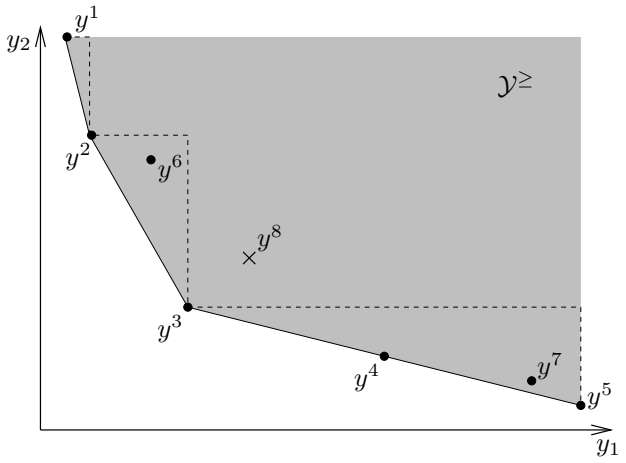


Figure 2: The Criterion Space.

Consider the following biobjective minimization problem:

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

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

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

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

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

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 supported extreme nondominated points define a number of triangles in which unsupported nondominated points may be found, as can be seen in the illustration of the criterion space 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^4 is the only nonextreme. Unsupported nondominated points are y^6 and y^7, while y^8 is dominated.

For practical reasons, it may be enough to find an approximation of the nondominated set, and it may for some large problem sizes be too time-consuming to find all the nondominated criterion points. In such a case, the concepts of ε-domination and ε-approximation introduced by Warburton (1987) can be used to control the quality of the set of criterion points reported.

Definition 1 A point $y = (y_1, y_2)$ *ε-dominates* point $\hat{y} = (\hat{y}_1, \hat{y}_2)$ if $(1 - \varepsilon)y$ dominates $\hat{y}$, i.e., if $(1 - \varepsilon)y \leq \hat{y}$.

Definition 2 A set $\widetilde{\mathcal{Y}}$ is an *ε-approximation* of a nondominated set $\mathcal{Y}_N$ if, for each point $\hat{y} \in \mathcal{Y}_N$, there exists $y \in \widetilde{\mathcal{Y}}$ that ε-dominates it.

Note that by finding an ε-approximation, it is ensured that any nondominated point is kept within a prespecified range from the nearest point in the approximation.

It is well known that unsupported nondominated points may exist for BiAP (Ulungu and Teghem 1995, Ehrgott 2000), 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 (Ehrgott 2000, Serafini 1986). In particular, it can be shown along the lines given in Ehrgott (2000), that the set of supported nondominated points as well as the set of unsupported nondominated points in BiAP and hence in BiMMAP can be exponential in cardinality.

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 simplex pivot operation. Along the same lines as in Przybylski et al. (2006), 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 simplex 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. The worst-case computational complexity of the method is exponential, as is every exact method for solving the problem.

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, and in phase two supported nonextreme and unsupported nondominated points are found. Both phases make use of a parametric minimization problem. To define this problem, two supported nondominated points $y^1 = (y_1^1, y_2^1)$ and $y^2 = (y_1^2, y_2^2)$, with $y_1^1 < y_1^2$, are needed. During the two-phase method, two such points will always be available in both phases. The parametric function $f_\lambda(x)$ is defined as follows:

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

where $\lambda \in \mathbb{R}_+$ is defined by the slope of the line between y^1 and y^2:

$$\lambda = \lambda(y^1, y^2) := (y_2^1 - y_2^2)/(y_1^2 - y_1^1). \tag{5}$$

Let $x^\star$ denote an optimal solution of (4) for a given value of λ. It is well known that $(c^1 x^\star, c^2 x^\star)$ is a supported nondominated point.

Figure 3 shows the pseudo code for phase one. The procedure first finds the upper-left and the lower-right points (y^1 and y^5 in Figure 2). Given two supported extreme nondominated points y^+ and y^-, the *search direction* $\lambda = \lambda(y^+, y^-)$ defined by (5) is calculated and (4) is solved. If the optimal solution $x^\star$ corresponds to a new supported 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 supported extreme nondominated point has been found, and we proceed with the two next points in $\mathcal{S}$. The `Next` function with arguments $\mathcal{S}$ and y returns the point following y in $\mathcal{S}$. The procedure stops when no additional supported extreme nondominated points can be found.

Because there may exist supported nonextreme nondominated criterion points such as y^4 or unsupported nondominated criterion points such as y^6 in Figure 2, it is not in general possible to

```
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      S :={y^UL, y^LR};
6      y^+ :=y^UL; y^- :=y^LR;
7      while (y^+ ≠ y^LR) do
8          λ := λ(y^+, y^-);
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 S;
11         else y^+ :=y^-;
12         y^- :=Next(S, y^+);
13     end while
14 end procedure
```

Figure 3: Phase One – Finding Supported Extreme Nondominated Points.

```
1  procedure PhaseTwo(△(y^+, y^-))
2      λ := λ(y^+, y^-);
3      S :={y^+, y^-};
4      k:=1; LB:=λ y_1^+ + y_2^+; UB:=UpdateUB(S);
5      while (LB ≤ UB) do
6          y^k :=KBest(k, λ);
7          if (NonDom(y^k)) then
8              S :=S ∪ {y^k};
9              UB:=UpdateUB(S);
10         end if
11         LB:=λ y_1^k + y_2^k; k:=k+1;
12     end while
13 end procedure
```

Figure 4: Phase Two – Finding Unsupported Nondominated Points.

find all nondominated points during the first phase. These points are found in phase two, which searches each triangle defined by the set of supported extreme nondominated points found in phase one.

Consider a triangle $\triangle(y^+, y^-)$ defined by the supported extreme nondominated points y^+ and y^- and by the point (y_1^-, y_2^+). The second phase searches each triangle using a K-best procedure to rank the parametric weight $f_\lambda(x)$ in the *ranking direction* given by $\lambda = \lambda(y^+, y^-)$. 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 y is found inside the triangle, the upper bound is updated to $UB_1 = \max\{y_1^-\lambda + y_2;\ y_1\lambda + y_2^+\}$ (Ulungu and Teghem 1995).

A pseudo code for phase two is given in Figure 4, 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 kth 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.

$$\begin{aligned} \min\ \ & f_\lambda(x) = \left(\lambda c^1 + c^2\right) x = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{L_{ij}} (\lambda c^1_{ijl} + c^2_{ijl}) x_{ijl} \\ \text{s.t.}\ \ & x \in \mathcal{MMAP}. \end{aligned} \tag{6}$$

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

$$l^\star_{ij}(\lambda) = \underset{1\le l\le L_{ij}}{\arg\min}\{\lambda c^1_{ijl} + c^2_{ijl}\}\ . \tag{7}$$

That is, for a specific value of λ, the minimal parametric cost entry, $l^\star_{ij}(\lambda)$, in each assignment cell (i, j), is chosen. It is straightforward to see that (6) reduces to the following problem:

$$\begin{aligned} \min\ \ & f_\lambda(x) = \left(\lambda c^1 + c^2\right) x = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijl^\star_{ij}(\lambda)} x_{ijl^\star_{ij}(\lambda)} \\ \text{s.t.}\ \ & x \in \mathcal{AP}, \end{aligned} \tag{8}$$

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* $\min\{\sum_{ij} c_{ijl^\star_{ij}(\lambda)} x_{ijl^\star_{ij}(\lambda)}|x \in \mathcal{AP}\}$.

Therefore, the minimum-cost assignment for BiMMAP given a fixed value of λ can be found in procedure `PhaseOne` by solving a single criterion AP. Moreover, because 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 supported extreme nondominated points* y^+ *and* y^-*. Then, no further supported extreme nondominated points can be found below the line connecting* y^+ *and* y^- *in phase one, if any of the following conditions are fulfilled:*

$$y^-_1 - y^+_1 = 1 \quad \text{or} \quad y^+_2 - y^-_2 = 1\ . \tag{9}$$

That is, we simply skip the search between y^+ and y^- if (9) holds in phase one. Condition (9) may also be used in phase two to skip searching some triangles. If any of the conditions in (9) are fulfilled, we do not have to apply procedure `PhaseTwo` to triangle $\triangle(y^+, y^-)$ because 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 3) may reduce the computational effort in phase two.

Furthermore, because all nondominated points have integer coordinates, the upper bound used in phase two may be improved. The following result was discovered independently in Pedersen et al. (2005b) and in Przybylski et al. (2008).

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* $\lambda = \lambda(y^+, y^-)$*, define*

$$UB^{IP} = \max_{i=1,\ldots,q-1}\{\lambda\left(y^{i+1}_1 - 1\right) + \left(y^i_2 - 1\right)\}\ . \tag{10}$$

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

As a result, we can use (10) in function `UpdateUB` of procedure `PhaseTwo`. Furthermore, note that upper bound (10) is valid for all bicriterion problems with integer criterion points and is an improvement to the upper bounds reported in the literature (Ulungu and Teghem 1995, Tuyttens et al. 2000).

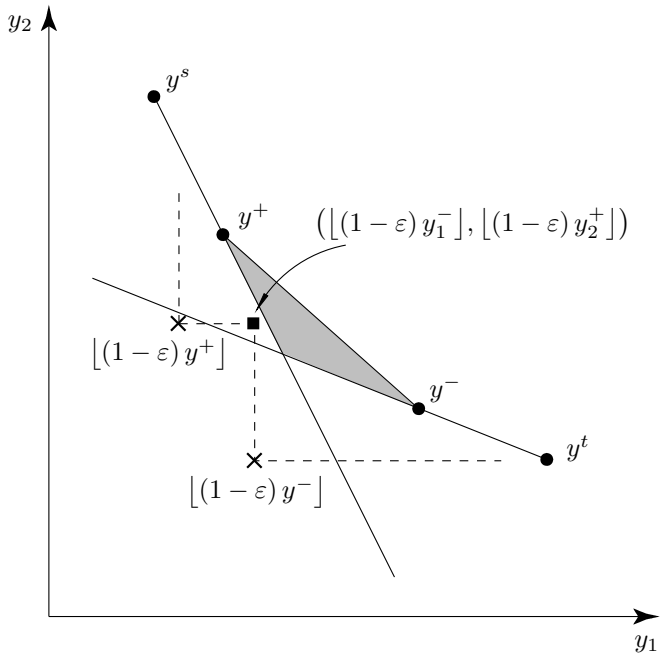


Figure 5: Using ε-Dominance in the First Phase.

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 phase two are needed.

First, consider phase one and a set of supported extreme nondominated points found during phase one as illustrated in Figure 5. Note that any new supported extreme nondominated point between y^+ and y^- must belong to the shaded area in Figure 5. As a result, we can skip searching for such 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 5, is ε-dominated by either y^+ or y^- if*

$$\begin{aligned} & \lambda_1 \lfloor(1-\varepsilon)\,y_1^-\rfloor + \lfloor(1-\varepsilon)\,y_2^+\rfloor \le \lambda_1 y_1^+ + y_2^+ \\ \textit{or} \quad & \lambda_2 \lfloor(1-\varepsilon)\,y_1^-\rfloor + \lfloor(1-\varepsilon)\,y_2^+\rfloor \le \lambda_2 y_1^- + y_2^- \quad , \end{aligned} \tag{11}$$

where $\lambda_1 = \lambda(y^s, y^+)$ and $\lambda_2 = \lambda(y^-, y^t)$.

Proof. We only show the result if the first condition in (11) is satisfied. The second case is shown with similar arguments. Therefore, assume that $\lambda_1 \lfloor(1-\varepsilon)\,y_1^-\rfloor + \lfloor(1-\varepsilon)\,y_2^+\rfloor \le \lambda_1 y_1^+ + y_2^+$. Furthermore, suppose that there exists a nondominated point (y_1, y_2) in the shaded area between y^+ and y^- (see Figure 5), 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 that

$$\lambda_1 y_1 + y_2 \le \lambda_1 \lfloor(1-\varepsilon)\,y_1^-\rfloor + \lfloor(1-\varepsilon)\,y_2^+\rfloor. \tag{12}$$

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

$$\lambda_1 \lfloor (1-\varepsilon)\, y_1^- \rfloor + \lfloor (1-\varepsilon)\, y_2^+ \rfloor \leq \lambda_1 y_1^+ + y_2^+ < \lambda_1 y_1 + y_2. \tag{13}$$

Combining equations (12) and (13), 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 can be used in procedure `PhaseOne` to skip the search between two points y^+ and y^-.

In phase two, the upper bound (10) can be further strengthened if an ε-approximation is wanted.

Proposition 5 *Given a 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*

$$UB^{IP}(\varepsilon) = \max_{i=1,\ldots,q-1} \{\lambda \lfloor (1-\varepsilon) y_1^{i+1} \rfloor + \lfloor (1-\varepsilon) y_2^i \rfloor\}\,, \tag{14}$$

where $\lambda = \lambda(y^+, y^-)$. 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 $\triangle(y^+, y^-)$. Therefore, there exists an $i \in \{1, \ldots, q-1\}$ such that y is located between y^i and y^{i+1}. If y is not ϵ-dominated, then

$$\begin{aligned}
& y_1 < (1-\varepsilon)\, y_1^{i+1} \;\wedge\; y_2 < (1-\varepsilon)\, y_2^i \\
\Rightarrow \quad & \text{(because } y \text{ is integral)} \; y_1 \leq \lfloor (1-\varepsilon)\, y_1^{i+1} \rfloor \;\wedge\; y_2 \leq \lfloor (1-\varepsilon)\, y_2^i \rfloor \\
\Rightarrow \quad & \lambda y_1 + y_2 \leq \lambda \lfloor (1-\varepsilon)\, y_1^{i+1} \rfloor + \lfloor (1-\varepsilon)\, y_2^i \rfloor
\end{aligned}$$

Because nondominated points can be located between any two consecutive points y^i and y^{i+1}, we obtain expression (14) for the upper bound. □

It follows that for $\varepsilon > 0$, equation (14) 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 5) not searched in phase one (i.e., satisfying (11)), 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.

There can also 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, because the optimal solution of (6) corresponds to a supported 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 Multimodal Assignments

In this section, we describe our method for ranking multimodal 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. Note that when searching a particular triangle, solutions outside that triangle may be found. The more often this happens, the slower is

the search in that particular triangle. However, by storing all nondominated solutions found, the search in other triangles may finish faster.

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 multimodal 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 multimodal assignments. In this paper, we use a branching technique which is an extension of the branching technique originally proposed by Murty (1968). 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

$$\mathcal{A}^1 = \{a \in \mathcal{A} \mid (1, j_1, l_1) \notin a\},$$
$$\mathcal{A}^i = \{a \in \mathcal{A} \mid (1, j_1, l_1), \ldots, (i-1, j_{i-1}, l_{i-1}) \in a, (i, j_i, l_i) \notin a\},\ i = 2, \ldots, n-1\ .$$

Clearly, the second-best assignment a_2 can be identified using a solution technique to find the minimum-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 that 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

$$\hat{a} = \{(1, j_1, l_1), (2, j_2, l_2), \ldots, (n, j_n, l_n)\}\ . \tag{15}$$

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:

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

Step 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 multimodal assignment problem defined by the two steps above. Due to Proposition 1, we have the following.

Corollary 2 *The minimum-cost multimodal assignment in MMAP($\hat{\mathcal{A}}^i$) can be found by solving a classical assignment problem, denoted AP($\hat{\mathcal{A}}^i$), using the minimum 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 in Pedersen et al. (2005a). The algorithm uses a reoptimization solution technique such that the minimum-cost assignments for the subsets can be found easily (see (Pedersen et al. 2005a) for more details). Because the general branching technique described above does not create more subsets than the classic branching technique, the overall complexity for ranking the K-best multimodal assignments is the same.

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

Actually, in some cases the minimum-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 (15) 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

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

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. Hence, we have the following proposition proved in Pedersen et al. (2005b).

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

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

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 minimum-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, because 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 Implementational Details

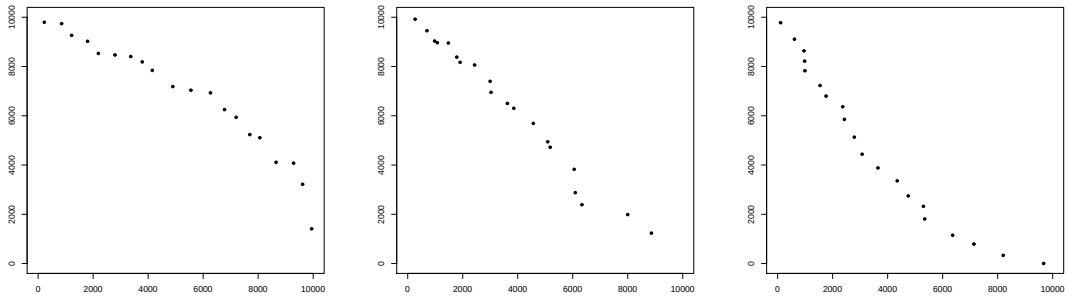
The algorithms have been implemented in C++ and compiled with the GNU C++ compiler version 3.4.5 using optimize option `-O3`.

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 (1987) in a slightly modified version allowing problems of varying sizes to be solved. Furthermore, we take advantage of Proposition 2 or Proposition 4 (if $\varepsilon > 0$) whenever possible.

In phase two, the parametric costs are again updated and ordered in nondecreasing order for a given ranking direction specified by λ. Next, the K-best multimodal assignment procedure described in Section 4 is utilized for searching a triangle, using the upper bounds given in Proposition 3 or Proposition 5 (if $\varepsilon > 0$).

The K-best multimodal assignment procedure was implemented using the reoptimization algorithm in Pedersen et al. (2005a) for ranking assignments for the classical AP, 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 entry in the array. That is, we just have to increase a local pointer by one to find the new minimal cost. For more details on the ranking implementation, see Pedersen et al. (2005a), an earlier version of Pedersen et al. (2007).



(a) shape −60 (b) shape 0 (c) shape 60

Figure 6: Cell Entries for Method 1.

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

5.2 BiMMAP Test Instances

The bicriterion multimodal assignment problem has, to the best of our knowledge, not previously been studied in the literature, and hence no available test instances exist for this problem. To facilitate a comprehensive computational study of our BiMMAP algorithm, a problem generator, *APGen*, was built for this problem class. As a side effect, our generator can be used to generate a variety of BiAP instances. The problem generator and the test instances used in this paper are downloadable from `http://www.research.relund.dk/`. 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 (Nielsen and Pedersen 2006). A BiMMAP instance is generated specifying a number of parameters:

n – size of the problem.

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

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

$maxCost$ – maximum-cost value (minimum-cost value is 0).

$method$ – 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, because this would correspond to a dominated solution. The number of entries in a cell is chosen randomly in the *entry range* $\{minEnt, \ldots, maxEnt\}$.

In Figure 6, we have displayed all two-dimensional cost vectors for a given cell having 20 entries generated by method 1. As can be seen 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 on the computational performance of our algorithm.

For method 2, the entries in a given cell are generated in a number of groups corresponding to the shape parameter. The groups are equally distributed in the cost space and the entries are

all generated fluctuating along the straight line between $(0, maxCost)$ and $(maxCost, 0)$. Note, to use method 2, the parameter $minEnt$ must be chosen sufficiently large, because at least two points are required in each group.

Finally, for method 3, the shape parameter has the same meaning as for method 1. However, here the cost space is divided into four regions by halving both criterion axes, and the entries are generated either in the upper-left cost region or in the lower-right cost region in consecutive cells. For graphical display of assignment cost cells generated using methods 2 and 3, see Nielsen and Pedersen (2006).

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

- $n \in \{4, 6, 8, 10\}$.
- Cost ranges : $\{0, \ldots, 500\}$ and $\{0, \ldots, 10,000\}$.
- 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 the number of entries are chosen to reflect a situation close to BiAP (few entries) and a situation very far away from BiAP (many entries), respectively. Note that the number of feasible assignments increases exponentially with the number of entries in each cell.

5.3 BiMMAP Exact Results

Let us first note that numerical studies showed the difficulty of the problem to be increasing in the size of the cost range and in the number of cell entries. The most significant factor is the entry range, obviously resulting from the increased number of feasible solutions (see (Pedersen et al. 2005b) for further details). Therefore, the focus in this section is entirely on problem instances using cost range $\{0, \ldots, 10,000\}$ and entry range $\{10, \ldots, 30\}$.

In Figure 7, we display, for the six different combinations of method and shape, the logarithm of the CPU time (in seconds) averaged over the 100 instances against problem size n. It can be seen that, for none of the six classes, the running time is increasing exponentially with problem size. Also note 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, it is important to note that phase two is the major time-consumer in this algorithm. More specifically, comparing the running times for all the $8,000$ exactly solved instances, phase two uses an average of 98% of the total CPU time. Consider Figure 8, where the nondominated points in the criterion space have been plotted for two test instances using method 1, shape -60 and method 1, 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. Note that these extreme points are far from each other, resulting in large triangles to search in the second phase. More important, all the supported extreme nondominated points are located almost on a straight line. Therefore, the ranking directions for the triangles are more or less the same. As a result, the ranking procedure of the time demanding phase two 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 that are outside the triangle currently searched are stored. This may enable the algorithm to finish searching other triangles faster, and hence enhance computational performance.

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

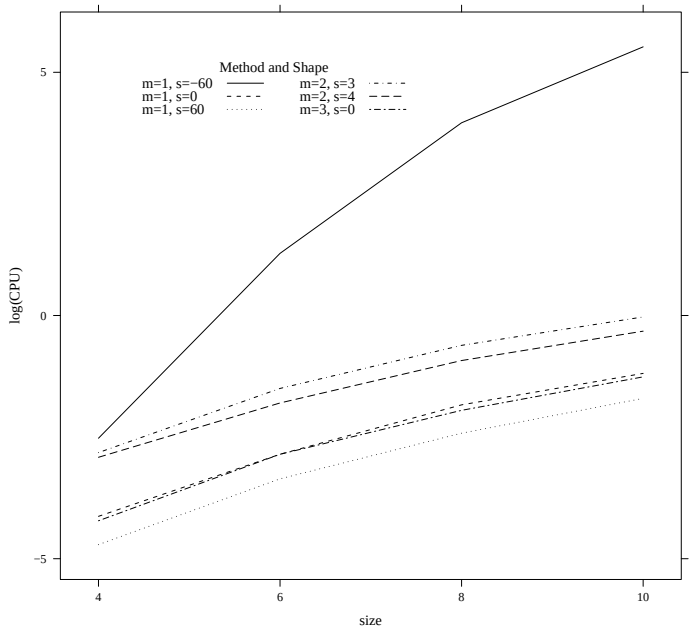
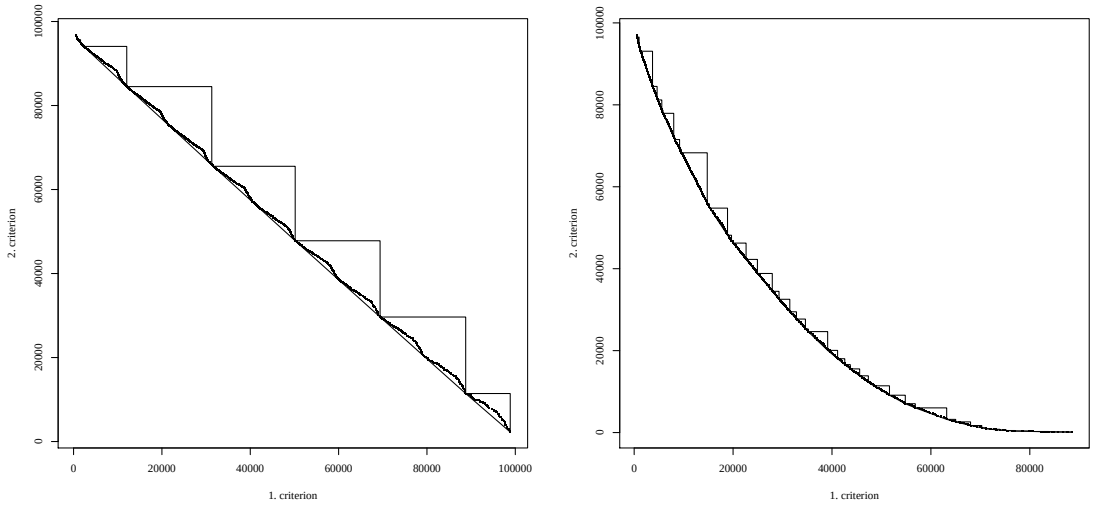


Figure 7: Logarithm of Average CPU Against n (Cost Range $\{0, \ldots, 10,000\}$ and Entry Range $\{10, \ldots, 30\}$).



(a) Shape −60

(b) Shape 60

Figure 8: Nondominated Points (Method 1, $n = 10$, Entry Range $\{10, \ldots, 30\}$, and Cost Range $\{0, \ldots, 10,000\}$).

Size	Avg. CPU	Max. CPU	Avg. SND	Avg. USND	Avg. ND
4	0.08	0.25	7	206	213
6	3.59	34.50	10	445	455
8	52.56	391.10	13	744	757
10	250.34	3412.34	16	1135	1151

Table 1: Exact Results (Method 1, Shape -60, Entry Range $\{10, ..., 30\}$, and Cost Range $\{0, ..., 10,000\}$).

	$\varepsilon = 0$			$\varepsilon = 0.01$			$\varepsilon = 0.05$		
Size	Avg.	90%	Max.	Avg.	90%	Max.	Avg.	90%	Max.
4	0.08	0.15	0.25	0.05	0.09	0.14	0.02	0.04	0.06
6	3.59	7.18	34.50	1.20	2.63	14.68	0.18	0.35	2.03
8	52.56	141.59	391.10	11.65	37.85	87.01	0.35	1.00	2.14
10	250.34	594.46	3412.34	34.73	82.90	468.09	0.28	0.63	3.94

Table 2: CPU Times for $\varepsilon = 0$, 0.01 or 0.05 (Method 1, Shape -60, Entry Range $\{10, ..., 30\}$, and Cost Range $\{0, ..., 10,000\}$).

Considering other instances, the above relationships proved to have general validity. Because method 1, shape -60 has established itself as the most difficult problem class, we focus on these instances only from here on. In Table 1, we give the numerical results for the exact solution of the instances. The first three columns depict the size of the problem, average CPU time (in seconds), and maximal CPU time, respectively. In the remaining three columns, we report average number of supported nondominated points, average number of unsupported nondominated points, and average of the total number of nondominated points, respectively. In general, we round the average number of nondominated points to the nearest integer. Obviously, all columns are increasing in size. However, it is interesting to note the relatively high number of unsupported nondominated criterion points making these instances very difficult.

5.4 BiMMAP Approximation Results

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 exactly solved instances ($\varepsilon = 0$).

In Figure 9, we graph the empirical cumulative distribution functions of CPU time for the 100 test instances of method 1, shape -60, entry range $\{10, \ldots, 30\}$, and cost range $\{0, \ldots, 10,000\}$ with size 10. 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. Figure 9 also clearly shows that the majority of 100 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 (in seconds), the 90% fractile of CPU time, and the maximum CPU time.

5.5 BiAP Test Results

Because BiMMAP is an extension of BiAP, we found it natural to test the performance of our current implementation on this problem class. Below, we summarize only a few of the obtained results. Readers requiring more information on these topics are referred to (Pedersen et al. 2005b).

To yield consistency in literature, we obtained the test instances used in (Tuyttens et al. 2000), which are BiAP instances of size $\{5, 10, \ldots, 50\}$. Also previously used in literature are BiAP instances of size $\{60, 70, \ldots, 100\}$ found in (Gandibleux et al. 2003). For all problem sizes, costs

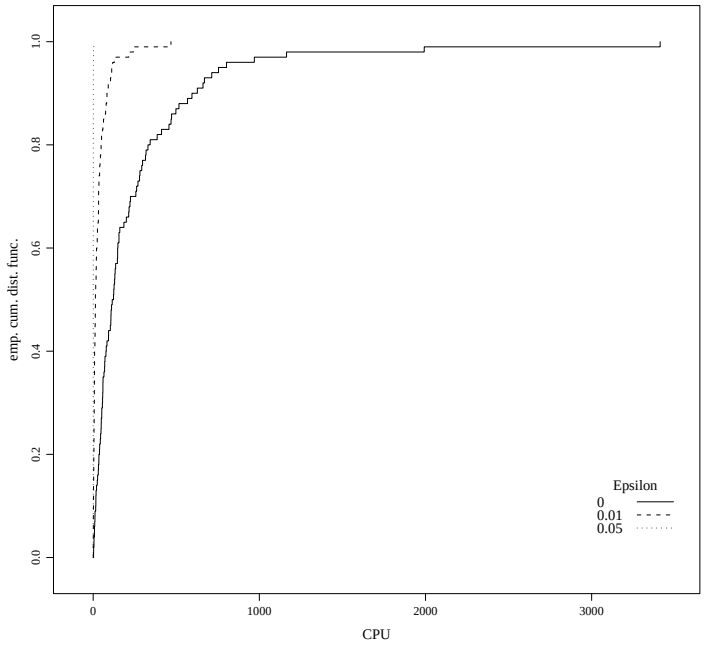


Figure 9: Empirical Cumulative Distribution of CPU time for Different ε Values (Method 1, Shape -60, Entry Range $\{10, \ldots, 30\}$, Cost Range $\{0, \ldots, 10,000\}$, and $n = 10$).

are chosen randomly in the rather narrow interval $\{0, \ldots, 19\}$, and only one instance of each size is available. By a comparison of CPU time, respecting the influence of the different computer architectures used, our algorithm was seen to outperform the exact methods previously reported in the literature.

To provide our reader with statistics based on a broader class of instances, we generated 100 instances, using APGen, of each of the following sizes $\{5, 10, \ldots, 100\}$ with costs randomly chosen in $\{0, \ldots, 1000\}$. This wide cost interval leaves room for identifying a large number of large triangles to search in phase two, and hence increase the 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 $\{0, \ldots, 1000\}$. For plots showing the difference in costs generated randomly and negatively correlated see (Nielsen and Pedersen 2006).

Figure 10 shows average CPU time against size for these BiAP test instances. For the negatively correlated instances, the algorithm was only capable of solving instances of problem size up to 40 within a reasonable amount of time. Therefore, only problems of size up to 40 are considered for the negatively correlated instances. This shows the complex nature of such instances, as is also previously seen for other bicriterion problems (see, for example, (da Silva et al. 2006)). The increased difficulty follows mainly because we have more and larger triangles to search in the second phase. Having CPU times no larger than 172 seconds for the random instances and 2455 seconds for the negatively correlated instances, our algorithm proves capable of solving BiAP problems rather efficiently.

References

da Silva, C. Gomes, J. Clímaco, J. Figueira. 2006. A scatter search method for bi-criteria $\{0, 1\}$-knapsack problems. *European Journal of Operational Research* **169** 373–391.

Ehrgott, M. 2000. *Multicriteria optimization*, *Lecture Notes in Economics and Mathematical Systems*, vol. 491. Springer-Verlag, Berlin.

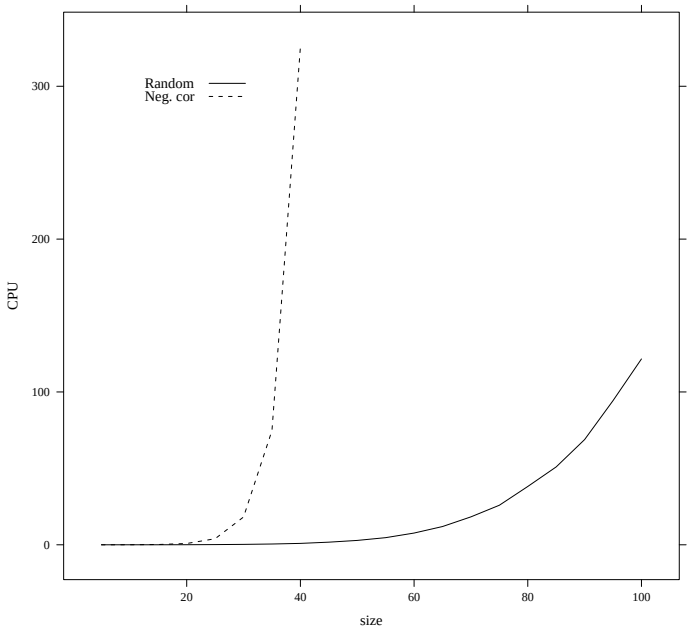


Figure 10: Average CPU Time Against n for Our BiAP Instances.

Gandibleux, X., H. Morita, N. Katoh. 2003. Use of genetic heritage for solving the assignment problem with two objectives. C.M. et al. Fonseca, ed., *Lecture Notes in Computer Science*, vol. 2632. Springer, Berlin/Heidelberg, 43–57.

Gandibleux, X., H. Morita, N. Katoh. 2005. A population-based metaheuristic for solving assignment problems with two objectives. *Journal of Mathematical Modelling and Algorithms.* Forthcoming.

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

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

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

Nielsen, L.R., K.A. Andersen, D. Pretolani. 2003. Bicriterion shortest hyperpaths in random time-dependent networks. *IMA Journal of Management Mathematics* **14** 271–303.

Nielsen, L.R., C.R. Pedersen. 2006. APGen - an assignment problem generator. Reference manual, University of Aarhus, Aarhus, Denmark. http://www.research.relund.dk.

Pedersen, C.R. 2006. Multicriteria Discrete Optimization – and Related Topics. Ph.D. thesis, Department of Operations Research, University of Aarhus, Aarhus, Denmark.

Pedersen, C.R., L.R. Nielsen, K.A. Andersen. 2005a. A note on ranking assignments using reoptimization. Working paper No. 2005/2, Department of Operations Research, University of Aarhus, Aarhus, Denmark. http://www.imf.au.dk/publs?id=585.

Pedersen, C.R., L.R. Nielsen, K.A. Andersen. 2005b. On the bicriterion multi modal assignment problem. Working paper No. 2005/3, Department of Operations Research, University of Aarhus, Aarhus, Denmark. http://www.imf.au.dk/publs?id=586.

Pedersen, C.R., L.R. Nielsen, K.A. Andersen. 2007. An algorithm for ranking assignments using reoptimization. *Computers and Operations Research.* doi:10.1016/j.cor.2007.04.008. Forthcoming.

Przybylski, A., X. Gandibleux, M. Ehrgott. 2006. The biobjective integer minimum cost flow problem - incorrectness of Sedeño-Noda and González-Martín's algorithm. *Computers and Operations Research* **33** 1459–1463.

Przybylski, A., X. Gandibleux, M. Ehrgott. 2008. Two phase algorithms for the biobjective assignment problem. *European Journal of Operational Research* **185** 509–533.

Serafini, P. 1986. Some considerations about computational complexity for multi objective combinatorial problems. J. Jahn, W. Krabs, eds., *Recent Advances and Historical Development of Vector Optimization*, *Lecture Notes in Economics and Mathematical Systems*, vol. 294. Springer, Berlin, 222–232.

Tuyttens, D., J. Teghem, Ph. Fortemps, K. Van Nieuwenhuyze. 2000. Performance of the MOSA method for the bicriteria assignment problem. *Journal of Heuristics* **6** 295–310.

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

Warburton, A. 1987. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research* **35** 70–79.