Algorithmic Operations Research

A two-phase method for bi-objective combinatorial optimization and its application to the TSP with profits

Carlo Filippi and Elisa Stevanato

Volume 7, Number 2, Winter 2012

URI: https://id.erudit.org/iderudit/aor7_2art08

See table of contents

Publisher(s)

Preeminent Academic Facets Inc.

ISSN

1718-3235 (digital)

Explore this journal

Cite this article

Filippi, C. & Stevanato, E. (2012). A two-phase method for bi-objective combinatorial optimization and its application to the TSP with profits. *Algorithmic Operations Research*, *7*(2), 125–139.

Article abstract

We study a variant of the two-phase method for general bi-objective combinatorial optimization problems. First, we analyze a basic enumerative procedure, often used in literature to solve specific bi-objective combinatorial optimization problems, making it suitable to solve general problems. We show that the procedure generates the exact set E *of efficient points by solving exactly* 2[notdef]E[notdef] − 1 *single objective problems. Second, we embed the procedure in a classic two-phase framework, where supported points are computed in the first phase and unsupported points are computed in the second phase.*

We test the refined approach on a hard problem, namely the Traveling Salesman Problem with Profits, a bi-objective generalization of the well known Traveling Salesman Problem. On the tested instances, the procedure outperforms the [epsilon1]*-constraint method, one of the most used approaches to solve exactly general bi-objective combinatorial optimization problems.*

Algorithmic Operations Research Vol.7 (2013) 125–139

A two-phase method for bi-objective combinatorial optimization and its application to the TSP with profits

Carlo Filippi [a]

[a]Department of Economics and Management, University of Brescia, Contrada S. Chiara 50, 25122 Brescia, Italy

Elisa Stevanato [b]

[b]Department of Economics and Management, University of Brescia, Contrada S. Chiara 50, 25122 Brescia, Italy

Abstract

We study a variant of the two-phase method for general bi-objective combinatorial optimization problems. First, we analyze a basic enumerative procedure, often used in literature to solve specific bi-objective combinatorial optimization problems, making it suitable to solve general problems. We show that the procedure generates the exact set $\mathcal{E}$ of efficient points by solving exactly $2|\mathcal{E}| - 1$ single objective problems. Second, we embed the procedure in a classic two-phase framework, where supported points are computed in the first phase and unsupported points are computed in the second phase.

We test the refined approach on a hard problem, namely the Traveling Salesman Problem with Profits, a bi-objective generalization of the well known Traveling Salesman Problem. On the tested instances, the procedure outperforms the ϵ-constraint method, one of the most used approaches to solve exactly general bi-objective combinatorial optimization problems.

Key words: Combinatorial optimization, Multiple objective programming, Efficient points, Pareto optima, Two-phase method, Traveling salesman with profits

1. Introduction

Decision makers often have to deal with several, usually conflicting, objectives. Often, multiple criteria are aggregated into a single objective function, but this simplification requires the adoption of arbitrary rules, that are usually not adequate to face the complexity of real world decisions. Multi-criteria decision making allows a degree of freedom which is lacking in more popular single objective optimization, as it aims to simultaneously optimize two or more conflicting objectives subject to certain constraints. [20] described how manufacturing firms are forced to consider multiple criteria when designing products in order to stay competitive. Focusing on a single objective, such as minimizing cost, is no longer sufficient in different markets. Several objectives need to be prioritized and balanced ([20]). Accordingly, in the service industry, limiting the objective to the minimization of costs may not be sufficient. In the long term, customer service and customer satisfaction yield a more profound basis for a healthy business, and it might be worthwhile to cut down on the number of customers while improving service quality. Thus, a trade-off is made between quantity and quality, and a bi-criterion approach is interesting and relevant. These types of application explain the interest in multi-objective optimization, as witnessed by many books and surveys on such a topic (see, e.g., [6], [5], [13]).

Since many real world applications involve discrete decisions or events, many studies involve multi-objective combinatorial optimization (MOCO) problems. In this paper, we focus on a sub-class of MOCO problems, namely bi-objective combinatorial optimization (BOCO) problems, where two competing criteria must be simultaneously optimized. The general BOCO problem may be formulated as follows:

$$\begin{aligned} \text{(BP)} \qquad & \min\ f(x) \\ & \min\ g(x) \\ & \text{subject to } x \in X, \end{aligned}$$

Email: Carlo Filippi [filippi@eco.unibs.it], Elisa Stevanato [elisa.stevanato@eco.unibs.it].

where f and g are the two objectives to be minimized, and X is the feasible region defined by the constraints, including integrality requirements. Any vector $x \in X$ is a *feasible solution*. We assume that f and g are bounded and take integer values on X.

If a BOCO problem is well formed, there should not be a single solution that simultaneously minimizes both objectives to their fullest. A feasible solution x is *Pareto optimal* if there exists no feasible solution x' such that $f(x') \leq f(x)$ and $g(x') \leq g(x)$, and at least one inequality is strict. A feasible solution x is *weakly Pareto optimal* if there exists no feasible solution x'' such that $f(x'') < f(x)$ and $g(x'') < g(x)$. A point $(\phi, \gamma) \in \mathbb{R}^2$ is an *efficient point* (resp. a *weakly efficient point*) if there exists a Pareto optimal solution (resp. a weakly Pareto optimal solution) x such that $f(x) = \phi$ and $g(x) = \gamma$. Let $\mathcal{E}$ denote the set of efficient points. An efficient point $(\phi, \gamma) \in \mathbb{R}^2$ is *supported* if there exists a scalar $\lambda \in [0, 1]$ and a feasible solution x such that x minimizes $\lambda f(x) + (1 - \lambda)g(x)$ and $f(x) = \phi$, $g(x) = \gamma$. All the efficient points that do not belong to the support set are defined *unsupported efficient points*. A supported efficient point that is an extreme point of the convex hull of all efficient points, is called an *extreme* supported efficient point. Let $\mathcal{SE}$ denote the set of extreme supported efficient points. Clearly, $\mathcal{SE} \subseteq \mathcal{E}$.

A general goal in BOCO is to find all efficient points and, for each of them, a corresponding Pareto optimal solution. Since the cardinality of the efficient set may grow exponentially with respect to the input size, often the efficient set is described only approximately. However, when the size of the efficient set is not huge, providing an exact description of the solutions is a reasonable task. In this paper we focus on such a situation.

We refer to [6] for an extensive literature review on BOCO methods. Here, we recall the approaches most strictly related to our work. The most popular method for BOCO problems is *weighted sum scalarization*, where the two objective functions are combined into a parametrized single criterion, leading to the following parametric single objective combinatorial optimization problem:

$$(P(\lambda)) \qquad \begin{aligned} \min\ & \lambda f(x) + (1 - \lambda)g(x) \\ \text{subject to}\ & x \in X. \end{aligned}$$

By varying the value of the parameter λ between 0 and 1, the method finds all extreme supported efficient points (see, e.g., [4] and [1]). The main disadvantage of this approach is that it cannot find the unsupported efficient points, that usually are the main part of the efficient set in BOCO problems. This drawback may be overcome by a *two-phase method*, where in the first phase all supported efficient points are found through a weighted sum scalarization, while in the second phase unsupported points are found by problem-specific methods. The two-phase method has been widely used. Among the most recent works, [19] presented a two-phase algorithm with several improvements to solve the bi-objective assignment problem, and later generalized it to multi-objective integer programs with more than two objectives ([17]). [21] developed a two-phase procedure to solve the bi-objective integer minimum cost flow problem.

Another popular approach to BOCO problems is the *ϵ-constraint method*. Here, one objective function is retained as a scalar-valued objective while the other generates a new constraint. The right-hand side of this constraint is denoted by ϵ: monotonically varying it, the whole set of efficient points can be generated. [2] revisited the ϵ-constraint method for BOCO problems, and test empirically its performance on a hard benchmark problem, namely the bi-objective Traveling Salesman Problem with Profits (TSPP).

Finally, a basic enumerative approach has been used in the literature to solve specific combinatorial optimization problems. At each iteration, the procedure combines linearly the objective functions as in weighted sum scalarization, and solves a single objective optimization problem by constraining the objective values in order to search an area strictly included between two already known efficient points. For instance, [9] used this approach to compute the efficient set for the assignment problem, while [14] suggested and implement the method specifically for the bi-objective Traveling Purchaser Problem.

In this paper, we formalize and analyze the basic enumerative approach, called here *BE algorithm*, in a general BOCO contex. In particular, we show that the BE algorithm generates the exact set $\mathcal{E}$ of efficient points by solving exactly $2|\mathcal{E}| - 1$ single objective problems. We also embed it in a general two-phase procedure. By the way it is designed, the BE algorithm is particularly suitable for cases where problem $(P(\lambda))$ is hard, and a branch-and-cut method is a reasonable choice for its solution. For this reason, we test the effectiveness of the obtained two-phase method on the TSPP, using both randomly generated instances and instances taken from TSPLIB. To obtain a fair evaluation, we compare the performance of the two-phase BE algorithm with that of the ϵ-constraint method, the approach used by [2] for

the same problem. We implement some variants of both the BE algorithm and the ϵ-constraint algorithm for the TSPP, using the same architectural choices. It turns out that the two-phase BE algorithms outperforms on most instances the ϵ-constraint algorithm and it is more stable in its behavior.

The contribution of the paper is thus twofold. First, we formalize a general approach for the second phase of a two-phase algorithm for general BOCO problems. Second, we apply for the first time a two-phase approach to the TSPP, improving performance with respect to a state-of-the-art algorithm.

The paper is organized as follows. In Section 2. we describe the BE algorithm for general BOCO problems and we analyze its correctness and complexity. In Section 3. we embed the BE algorithm in a two-phase framework. In Section 4. we give an overview of the ϵ-constraint method. In Section 5. we define the TSPP, showing how the described algorithms can be implemented to solve it, and we present our computational results. Finally, some conclusions are drawn in Section 6.

2. An exact method for BOCO problems

In this section, we describe and analyze a general procedure for the computation of the efficient set of BOCO problems.

The underlying idea is to use the weighting method with additional constraints. The procedure is contained in a binary search that explores specific regions of the objective space, through the use of constraints that restrict the search to properly defined sub-areas. This allows us to generate both supported and unsupported points.

Let a general BOCO problem (BP) be given. The starting step of the algorithm computes the *ideal point* (ϕ^I, γ^I) defined by

$$\phi^I = \min_{x \in X} f(x), \quad \gamma^I = \min_{x \in X} g(x),$$

and the *Nadir point* (ϕ^N, γ^N) defined as

$$\phi^N = \min_{x \in X} \{ f(x) \,|\, g(x) = \gamma^I \},$$

$$\gamma^N = \min_{x \in X} \{ g(x) \,|\, f(x) = \phi^I \}.$$

Such points define lower and upper bounds on the coordinates of efficient points. By construction, the following two points are efficient:

$$(\phi^I, \gamma^N), \ (\phi^N, \gamma^I).$$

Furthermore, every efficient point is contained in the rectangle in the objective space having (ϕ^I, γ^N) as upper left corner and (ϕ^N, γ^I) as lower right corner. Thus, the algorithm initializes its partial list E of efficient points as $E = \{(\phi^I, \gamma^N), (\phi^N, \gamma^I)\}$, and its list L of pending search areas as $L = \{[(\phi^I, \gamma^N), (\phi^N, \gamma^I)]\}$, where a rectangular area is identified by its upper left and lower right corners respectively, in square brackets.

The algorithm iterates on areas contained in L, searching for efficient points. To do this, it solves a related single objective problem and gives (if it exists) a new efficient point to put in set E and two new areas to be added to L. The algorithm continues until all areas in L have been successfully explored.

During a generic iteration, let $[(\phi_1, \gamma_1), (\phi_2, \gamma_2)]$ be the area in the objective space picked up from L, with $\phi_1 < \phi_2$ and $\gamma_1 > \gamma_2$. Then, the single objective problem related to area $[(\phi_1, \gamma_1), (\phi_2, \gamma_2)]$ has the following formulation:

$$\begin{array}{lrl} (P(\lambda, \phi_2, \gamma_1)) & \min & \lambda f(x) + (1-\lambda) g(x) \\ & \text{subject to} & x \in X \\ & & f(x) \le \phi_2 - 1 \\ & & g(x) \le \gamma_1 - 1 \end{array}$$

where

$$\lambda = \frac{\gamma_1 - \gamma_2}{\gamma_1 - \gamma_2 + \phi_2 - \phi_1}. \tag{1}$$

Notice that, in the objective space, vector $(\lambda, 1-\lambda)$ is orthogonal to the line passing through (ϕ_1, γ_1) and (ϕ_2, γ_2) (see Figure 1(a)). If $(P(\lambda, \phi_2, \gamma_1)))$ has an optimal solution x^*, then x^* corresponds to a new efficient point $(f(x^*), g(x^*))$ to be inserted in L (see Figure 1(b)).

We call the whole procedure *BE algorithm*; the pseudo-code is given as Procedure 1. Note that the initial efficient points can be found without computing explicitly the ideal and the Nadir points. More precisely, (ϕ^I, γ^N) and (ϕ^N, γ^I) correspond to the optimal solution of $P(1-\varepsilon, \Phi, \Gamma)$ and $P(\varepsilon, \Phi, \Gamma)$, respectively, where ε is a small positive constant, and Φ and Γ are upper bounds on the values of f and g on X. Note further that we do not specify the search strategy related to the data structure used for list L, since such a strategy does not affect the total number of iterations.

The BE algorithm has not been analyzed previously, so we prove the following result.

Theorem 1 *The BE algorithm is correct. The number of problems* $(P(\lambda, \phi_2, \gamma_1)))$ *that have to be solved is equal to* $2|\mathcal{E}| - 1$.

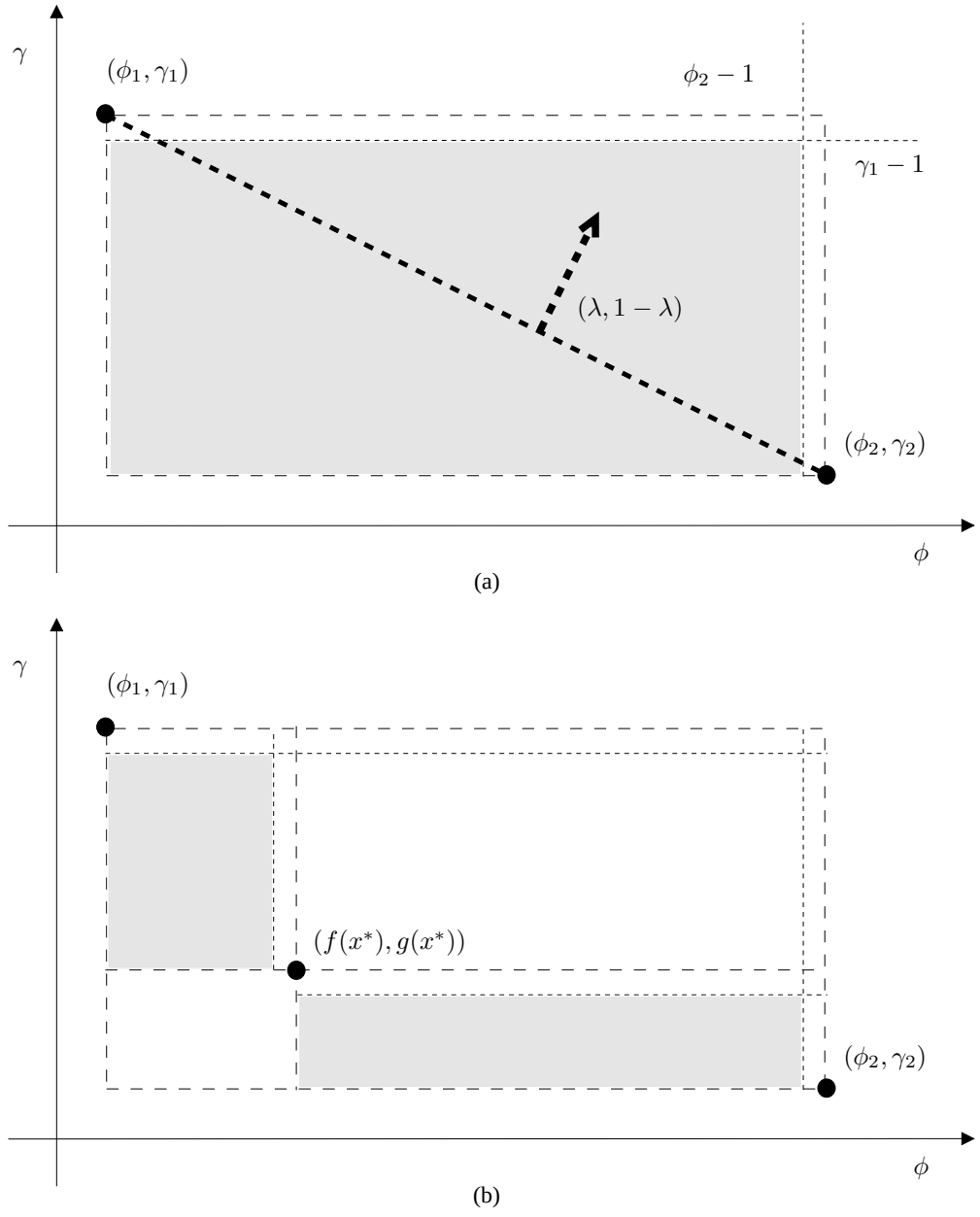

Fig. 1. Problem $(P(\lambda, \phi_2, \gamma_1))$: (a) objective function (bold dashed) and search area (grey); (b) insertion of a new efficient point and two new search areas.

Algorithm 1 BE algorithm

solve problem $P(1-\varepsilon,\Phi,\Gamma)$; let x^* be the returned optimal solution
$(\phi^I,\gamma^N)=(f(x^*),g(x^*))$
solve problem $P(\varepsilon,\Phi,\Gamma)$; let x^* be the returned optimal solution
$(\phi^N,\gamma^I)=(f(x^*),g(x^*))$
$E = \{(\phi^I,\gamma^N),\ (\phi^N,\gamma^I)\}$, $L = \{[(\phi^I,\gamma^N),(\phi^N,\gamma^I)]\}$
while $L\neq\emptyset$ **do**
  remove one interval $[(\phi_1,\gamma_1),(\phi_2,\gamma_2)]$ from L
  $\lambda=(\gamma_1-\gamma_2)/(\gamma_1-\gamma_2+\phi_2-\phi_1)$
  solve problem $(P(\lambda,\phi_2,\gamma_1))$
  if $(P(\lambda,\phi_2,\gamma_1)))$ is feasible **then**
    let x^* be the returned optimal solution
    insert $(f(x^*),g(x^*))$ in E, between (ϕ_1,γ_1) and (ϕ_2,γ_2)
    $L = L \cup \left\{[(\phi_1,\gamma_1),(f(x^*),g(x^*))],\ [(f(x^*),g(x^*)),(\phi_2,\gamma_2)]\right\}$
  end if
end while
return E

Proof To prove the correctness of the algorithm we show that:

a) Every time a problem $(P(\lambda,\phi_2,\gamma_1)))$ has an optimal solution, this identifies an efficient point.
b) Every efficient point is found by a problem $(P(\lambda,\phi_2,\gamma_1)))$.

We prove the above points by contradiction.

a) Let us suppose that a problem $(P(\lambda,\phi_2,\gamma_1)))$ identifies a point (ϕ^*,γ^*) which is not efficient, in the sub-area of the objective space delimited by the efficient points (ϕ_1,γ_1) and (ϕ_2,γ_2). The efficiency hypothesis on these points excludes that there may be other efficient points in the areas located in the lower-left side of them. This means that, in the sub-area defined by (ϕ_1,γ_1) and (ϕ_2,γ_2), there is a point (ϕ,γ) that dominates (ϕ^*,γ^*). It follows that $\phi\leq\phi^*$ and $\gamma\leq\gamma^*$, and at least one inequality holds strictly. Since $0<\lambda<1$,

$$\lambda\phi+(1-\lambda)\gamma<\lambda\phi^*+(1-\lambda)\gamma^*,$$

but this contradicts the fact that (ϕ^*,γ^*) corresponds to the optimal solution returned by problem $(P(\lambda,\phi_2,\gamma_1)))$.

b) Let us suppose that there is an efficient point (ϕ,γ) not found by the BE algorithm. This point must be located in the sub-area of the objective space delimited by two consecutive efficient points (ϕ_1,γ_1) and (ϕ_2,γ_2), contained in the final set E. It means that the search area $[(\phi_1,\gamma_1),(\phi_2,\gamma_2)]$ has been considered by the BE algorithm, and the corresponding problem $(P(\lambda,\phi_2,\gamma_1)))$ returned no solution. This proves the contradiction.

Starting from the above statements, it is easy to compute the exact number of problems $(P(\lambda,\phi_2,\gamma_1)))$ that must be solved to obtain the entire exact Pareto optimal set. The first 2 efficient points are found in the initialization phase, where 2 special problems $(P(\lambda,\phi_2,\gamma_1)))$ are solved. The other $|\mathcal{E}|-2$ points are computed by $|\mathcal{E}|-2$ problems $(P(\lambda,\phi_2,\gamma_1)))$. To check that no efficient points are contained in the final intervals we need exactly $|\mathcal{E}|-1$ calls to problem $(P(\lambda,\phi_2,\gamma_1)))$. The statement follows. □

Note that the bound given by the theorem is tight and does not depend on the adopted search strategy. The bound implies that the complexity of the BE algorithm is polynomial at least in the output size whenever each problem $(P(\lambda,\phi_2,\gamma_1)))$ is polynomially solvable. Alternatively, the bound might be used to guarantee that only a polynomial number of calls to problem $(P(\lambda,\phi_2,\gamma_1)))$ are necessary if the objective functions have special characteristics (see, e.g., [15] for a discussion in case of multi-objective shortest path problems).

Remark 2 *Theorem 1 does not depend on the specific value of λ used in problem $(P(\lambda,\phi_2,\gamma_1)))$, provided $0<\lambda<1$. Therefore, we can choose the λ value in order to improve numerical stability. The most reasonable choice is to fix $\lambda=\frac{1}{2}$. In this way the objective function of the problem $(P(\lambda,\phi_2,\gamma_1)))$ is equivalent to the sum of the single objectives, which always take integer values on the feasible region X. We refer to the BE algorithm where the λ value is always fixed to $\frac{1}{2}$ and the objective doubled as* FBE algorithm.

We will compare through computational test the effectiveness of the BE and FBE algorithms.

3. A two-phase method

At each iteration of the BE algorithm we solve a problem $(P(\lambda,\phi_2,\gamma_1)))$, which has two more constraints than the original BOCO problem. These additional constraints may bring about an increment of the computational time needed to solve each problem. In order to reduce this phenomenon, we may use the BE algorithm as a sub-routine of a two-phase method, where supported and unsupported efficient points are computed in two

different phases. In this way, we may adopt tools optimized for the different types of problem, improving the overall performance of the method.

The first phase focuses on the computation of supported efficient points. To implement a classical first phase, we simply apply the BE algorithm with the following modification: at each iteration of the algorithm, solve a problem $(P(\lambda))$ instead of $(P(\lambda, \phi_2, \gamma_1)))$, i.e., remove the constraints defining a specific search area. The advantage is having lighter subproblems to solve, while the disadvantage is that we may end up with an already known efficient point, namely an extreme point of the considered rectangular area.

It is interesting to note that [14] suggest an opposite approach, where each subproblem is even more constrained. Indeed, they propose to maintain the two constraints and add a third one:

$$\lambda f(x) + (1-\lambda) g(x) < \lambda \phi_2 + (1-\lambda)\gamma_1.$$

In this way, we never end up with an already known point (if the current interval does not contain an extreme supported point then the subproblem turns out to be infeasible). However, a clear disadvantage is that the subproblem is heavier to solve. Some computational tests will confirm that Salazar and Ledesma's approach is not competitive in practice.

In the second phase we compute the unsupported points by searching iteratively new solutions on all the subareas defined by couples of consecutive supported points computed in the first phase. More precisely, we call either the BE or the FBE algorithm, without initialization steps, starting from every couple of consecutive points in the list E by the first phase.

3.1. *Upper bounds for the second phase*

The performance of the BE or FBE algorithm during the second phase could be improved by using upper bounds of the objective function of problem $(P(\lambda, \phi_2, \gamma_1)))$.

Let (ϕ_0, γ_0) and (ϕ_q, γ_q) be two consecutive supported efficient points returned by the first phase. Let $(\phi_1, \gamma_1), (\phi_2, \gamma_2), \ldots, (\phi_{q-1}, \gamma_{q-1})$ be points (not necessarily efficient) corresponding to feasible solutions $x^1, x^2, \ldots, x^{q-1} \in X$ such that

$$\phi_0 < \phi_1 < \ldots < \phi_{q-1} < \phi_q \text{ and}$$

$$\gamma_0 > \gamma_1 > \ldots > \gamma_{q-1} > \gamma_q.$$

[23] observed that every efficient point (ϕ^*, γ^*) in $[(\phi_0, \gamma_0), (\phi_q, \gamma_q)]$ must satisfy

$$\lambda \phi^* + (1-\lambda)\gamma^* \leq \max_{i=1,\ldots,q} \{\lambda \phi_i + (1-\lambda)\gamma_{i-1}\},$$

where λ is the coefficient associated with problem $(P(\lambda, \phi_q, \gamma_0)))$. [18] observed that when the objective functions have integer values the bound can be improved as follows

$$\lambda \phi^* + (1-\lambda)\gamma^* \leq UB =$$

$$\max_{i=1,\ldots,q} \{\lambda(\phi_i - 1) + (1-\lambda)(\gamma_{i-1} - 1)\}.$$

Bound UB may be useful to speed up the resolution of $(P(\lambda, \phi_2, \gamma_1)))$ when branch-and-bound methods are used.

Remark 3 *The upper bound UB is referred to a value of λ defined as in (1). However, the bound is still valid if we use it in connection with the FBE algorithm, fixing $\lambda = \frac{1}{2}$ and doubling the result. More precisely, the FBE algorithm may be improved by means of the following, integer bound:*

$$FUB = \max_{i=1,\ldots,q} \{\phi_i + \gamma_{i-1} - 2\}.$$

4. The ϵ-constraint method

In order to test the practical efficiency of the BE algorithm, we will compare its performance with the ϵ-*constraint method*, that we briefly recall in this section.

In the ϵ-constraint method one objective function of (BP) is retained as a scalar-valued objective while the other objective function generates a new constraint. Thus, we may formulate two different subproblems:

$$(P_1(\epsilon)) \qquad \min_{x \in X} \{f(x) \,|\, g(x) \leq \epsilon\}$$

$$(P_2(\epsilon)) \qquad \min_{x \in X} \{g(x) \,|\, f(x) \leq \epsilon\}$$

One may use either problem $(P_1(\epsilon))$ or problem $(P_2(\epsilon))$. Starting from a sufficiently large value of ϵ and progressively reducing it, the exact Pareto optimal set can be generated. [2] analyzed the implementation of the ϵ-constraint method for BOCO problems with integer objective values. We report the basic method in Procedure 2, where we may set $\Delta = 1$. It is important to note that the above algorithm can easily be restated by swapping the role of f and g, thus solving problems $P_2(\epsilon)$.

A drawback of the ϵ-constraint method is that it may generate points that are not efficient, but only weakly

efficient. [16] proposed a novel version of the method, called *augmented ϵ-constraint method*, that avoids the production of weakly efficient points and accelerates the whole process by avoiding redundant iterations. It consists in the transformation of the objective function constraints to equalities by explicitly incorporating the appropriate slack or surplus variables. In the same time, these slack or surplus variables are used as a second term (with lower priority in a lexicographic manner) in the objective function, forcing the program to produce only efficient solutions. For instance, the following model is used in place of ($P_1(\epsilon)$):

$$(\overline{P}_1(\epsilon)) \qquad \min_{x\in X, s\geq 0}\{f(x)+\delta s \,|\, g(x)+s=\epsilon\}$$

where $\delta > 0$ is sufficiently small.

It turns out that the augmented method gives a relevant improvement with respect to the basic method when the number of objective functions is high (more than three) and/or when the size of the non efficient points computed by the basic method is considerable. As a consequence, for the specific class of problems considered in this work, the augmented method does not provide a clear advantage with respect to the basic ϵ-constraint method. Moreover, the introduction of the δ factor in the objective function may cause numerical stability issues. For these reasons, we decided to report only the basic ϵ-constraint method and remove possible dominated points through a straightforward post-processing.

Algorithm 2 ϵ-constraint algorithm

solve problem $P(1-\varepsilon, \Phi, \Gamma)$; let x^* be the returned optimal solution
$(\phi^I, \gamma^N) = (f(x^*), g(x^*))$
$\gamma^I = \min_{x\in X} g(x)$
$E = \{(\phi^I, \gamma^N)\}$, $\epsilon = \gamma^N - \Delta$
while $\epsilon \geq \gamma^I$ **do**
  compute an optimal solution x^* of $P_1(\epsilon)$
  add $(f(x^*), g(x^*))$ to E and set $\epsilon = g(x^*) - \Delta$
end while
return E

5. An Application: TSPP

In this section, we test the BE algorithm on the Traveling Salesman Problem with Profits (TSPP). First of all, we give a description and a formulation of the problem, and we show how the BE algorithm can be tailored on it. Then, we describe the test instances and the obtained results, comparing them with the performance of the ϵ-constraint method.

5.1. *Problem description*

The TSPP can be described as follows: let $G = (V, E)$ be a complete undirected graph, with $V = \{0, 1, \ldots, n\}$ the set of nodes, and E the set of edges. Let p_i be the profit of node i, and let c_e be the cost of edge $e \in E$. A service carrier, starting from node 0 may visit any subset of nodes. Visiting a subset S of nodes implies a profit $p(S) = \sum_{i\in S} p_i$, and a cost $c(S)$ which is the cost of a shortest Hamiltonian tour among the nodes in S. The carrier wishes to maximize collected profit and minimize incurred cost, and thus faces a bi-objective problem.

Some single objective variants of the TSPP have been considered in the literature as independent problems. If we look at the TSPP as a BOCO problem (BP), considering f as the cost function and $-g$ as the profit function, then problem ($P(\lambda)$) with fixed λ is known as the *profitable tour problem*, problem ($P_1(\epsilon)$) is known as the *Prize Collecting TSP* (PCTSP), and problem ($P_2(\epsilon)$) is known as the *Orienteering Problem* (OP). Recently, [3] studied different approaches to the TSPP on trees and other special graphs. We refer to [7] for a survey on the TSPP and related problems.

In order to formulate the TSPP, we use two sets of variables. The first one is associated with edges: $x_e = 1$ if edge e belongs to the solution, $x_e = 0$ otherwise ($e \in E$). The second one is associated with nodes: $y_i = 1$ if node i is visited, $y_i = 0$ otherwise ($i \in V$). We use the following model (cf. [7]):

$$\min \sum_{e\in E} c_e x_e, \quad \max \sum_{i\in V} p_i y_i \tag{2a}$$

$$\text{subject to} \sum_{e\in\delta(\{i\})} x_e = 2y_i \qquad (i \in V) \tag{2b}$$

$$\sum_{e\in\delta(S)} x_e \geq 2y_i \qquad (\emptyset \neq S \subset V, \text{ with } 0 \in V\setminus S \text{ and } i \in S) \tag{2c}$$

$$x_e \in \{0,1\} \qquad (e \in E) \tag{2d}$$

$$y_i \in \{0,1\} \qquad (i \in V) \tag{2e}$$

where $\delta(S)$ is the set of edges having exactly one endpoint in node set S.

Constraints (2b) are the *degree constraints*: they ensure that the degree of each node is 2 if the node is

visited, 0 if the node is not visited. This means that a node can be visited at most once. Constraints (2c) are the *subtour elimination constraints*: they impose that no subtours are allowed among visited nodes.

5.2. *Initialization*

In the TSPP, the computation of the first two efficient points is relatively easy. The ideal (minimum) cost is 0, corresponding to the empty tour, and the ideal (maximum) profit is $\sum_{i \in V} p_i$, corresponding to visiting all nodes. The Nadir value of profit given the ideal cost is 0, and the Nadir value of cost given the ideal value of profit is the minimum cost of a Hamiltonian tour on all nodes, say C^*. Hence, the starting points for the BE algorithm are

$$(\phi^I, \gamma^N) = (0, 0) \quad \text{and} \quad (\phi^N, \gamma^I) = (C^*, \sum_{i \in V} p_i).$$

5.3. *Solving* $P(\lambda, \phi_2, \gamma_1)$

If (BP) takes the form of (2), then problem $P(\lambda, \phi_2, \gamma_1)$ can be formulated as follows.

$$\min \lambda \sum_{e \in E} c_e x_e - (1 - \lambda) \sum_{i \in V} p_i y_i \tag{3a}$$

$$\text{subject to} \sum_{e \in \delta(\{i\})} x_e = 2y_i \qquad (i \in V) \tag{3b}$$

$$\sum_{e \in \delta(S)} x(e) \geq 2y_i \quad (\emptyset \neq S \subset V, \text{ with } 0 \in V \setminus S \text{ and } i \in S) \tag{3c}$$

$$\sum_{e \in E} c_e x_e \leq \phi_2 - 1 \tag{3d}$$

$$\sum_{i \in V} p_i y_i \geq \gamma_1 + 1 \tag{3e}$$

$$x_e \in \{0, 1\} \quad (e \in E) \tag{3f}$$

$$y_i \in \{0, 1\} \quad (i \in V) \tag{3g}$$

If we consider the FBE algorithm, the objective function becomes:

$$\min \sum_{e \in E} c_e x_e - \sum_{i \in V} p_i y_i$$

We solve problem (3) by a classic branch-and-cut procedure. Consider for simplicity the root node of the branch-and-cut tree. We start with no constraints (3c) and no integrality constraints, obtaining a (fractional) solution (x^*, y^*). To detect violated subtour elimination constraints, we set up a capacitated network by giving capacity x_e^* to edge e. We fix 0 as the source node and we consider as the destination each other node of the graph, iteratively. Then, we compute the minimum cut by a max-flow/min-cut algorithm. If the value of the max flow from 0 to i is smaller than $2y_i^*$ then the constraint (3c) associated with the corresponding minimum cut is violated, and we add it to the model. Once all cuts obtained in this way have been added, we re-optimize obtaining a new solution (x^*, y^*), where the process can be repeated.

5.3.1. *Warm start*

We use the Lin-Kernighan heuristic to suggest a feasible starting solution to each branch-and-cut search tree. In particular, we have tried a modified and extended version of the Lin-Kernighan algorithm, presented by [12]. We use Lin-Kernighan algorithm in the following way. We solve the relaxed problem (2) with no subtour elimination constraints and no integrality requirements. Let (x^*, y^*) be the obtained solution and let $G^* = (V, E^*)$, where $E^* = \{e \in E : x_e^* > 0\}$. We invoke Lin-Kernighan on the connected component of G^* containing node 0 and use the obtained subtour as starting feasible solution for problem (2).

5.4. *Computational results*

To assess the performance of the algorithms, we implemented a C++ code and ran it on an AMD Opteron 2.4 GHz processor equipped with CPLEX 12.1. To compute the cuts needed to solve each $P(\lambda, \phi_2, \gamma_1)$ problem, we used a function of the Concorde package [1]. The function is an implementation of the "push-relabel" flow algorithm described in [10]. For the Lin-Kernighan procedure, we use the implementation developed by [12] [2].

In order to have a fair comparison between the performances of the BE algorithm and the ϵ-constraint algorithm, we implemented both methods using the same libraries, the same functions and the same hardware resources. In this way, we obtained codes differing each other only for the description of each optimization model and the modality of exploration of the objective space. We implemented both variants of the ϵ-constraint method.

[1] http://www.tsp.gatech.edu/concorde.html

[2] http://www.akira.ruc.dk/∼keld/research/LKH/

5.4.1. *Randomly generated instances*

The first type of instances are randomly generated. Graph nodes are points generated in the square 1000×1000. The arc costs are the euclidean distance between each couple of points, rounded to the nearest integer value. Profits associated with each node belong to one of the following classes:

I. $p_i = 1$ for all $i \in V$;
II. p_i integer from a uniform distribution in the interval $[1, 5]$, for all $i \in V$;
III. p_i integer from a uniform distribution in the interval $[1, 20]$, for all $i \in V$.

For each class, we average the performances of the algorithms on a sample of 10 instances. The obtained results are contained in Table 1, Table 2, and Table 3. The informations reported in the columns are the following:

- $|V|$ is the number of nodes considered, including the source;
- $|\mathcal{SE}|$ is the number of supported efficient points;
- $|\mathcal{E}|$ is the number of efficient points;
- N is the number of computed points (dominated and non-dominated);
- $Time$ is the average CPU time, in seconds, to run the specified procedure.

We use the column identified by N to show the number of points in the objective space computed by the ϵ-constraint method. The difference $N - |\mathcal{E}|$ gives us the number of dominated points computed. Cells marked by "t.l.e." indicate that the instance was still unsolved after a time limit of 72 hours (259200 seconds). Cells marked by "*" indicate that, after the specified time, CPLEX exits the program for memory problems. In this case, we report in column $|\mathcal{E}|$ the number of efficient points computed before the exit. For each instance type the best average time is in bold.

Table 1 compares Ledesma and Salazar's implementation of the first phase (Constrained 1st Phase, cf. [14]) and our implementation (Unconstrained 1st Phase). Table 2 compares the computational times needed by the following algorithms:

- BE is the BE algorithm as described in Procedure 1;
- FBE is the FBE algorithm;
- $FBE + LK$ is the FBE algorithm with the Lin-Kernighan warm start;
- $2P$ is the two-phase method with the FBE algorithm in the second phase;
- $2P + FUB$ is the two-phase method with the FBE algorithm and the upper bound improvement in the second phase.

Note that we do not include the two-phase method with the BE algorithm and upper bound UB in the second phase, because this variant is systematically worse than $2P + FUB$.

Although the two-phase method with upper bounds does not prevail systematically, we can observe that, overall, it gives better results with respect to the other procedures. Thus, we retain the latter as our reference implementation and in Table 3 we compare it with both implementations of the ϵ-constraint method.

Note that the performance of the two-phase approach could be further improved by initializing the upper bounds through the heuristic generation of potentially efficient points, as in [18]. However, such a generation is beyond the scope of this evaluation.

5.4.2. *Instances from TSPLIB*

The second type of instances are TSP instances taken from TSPLIB ([22]). We generate profits according to [2], considering three types:

A. $p_i = 1$ for all $i \in V$;
B. $p_i = 1 + [(7141 \cdot i + 73) \mod 100]$ for all $i \in V$;
C. $p_i = 1 + \lceil 99 \cdot c_{0,i}/\theta \rceil$ for all $i \in V$, where $\theta = \max_{j \in V \setminus \{0\}} c_{0,j}$.

Instances with profits of type A are generally the easiest: the cardinality of the efficient frontier generated is equal to the number of nodes. Instances of type B have random profit values between 1 and 100, while instances of type C produce hard problems, where profit values become larger as their distance from the source increases.

Table 4 contains the times needed by the two implementations of the first phase to compute the set of supported efficient points. Table 5, Table 6, and Table 7 show the results obtained by running the different versions of the BE algorithm and two-phase method. Again, we observe that the two-phase procedure with the FBE algorithm and the upper bound improvement is the most efficient.

Finally, Table 8, Table 9, and Table 10 show the performance of the selected two-phase method, with respect to the ϵ-constraint approach both in the PCTSP and in the OP versions, with profit types A, B and C, respectively. Notice that for profit type A we have tested instances with up to 159 nodes, whereas for profit types B and C, producing harder instances, we have tested instances with up to 101 nodes.

5.4.3. *Discussion*

In the experiments we run, the times needed to compute the set of efficient points by the two-phase procedure with the FBE algorithm and the upper bound improvement are, in most cases, better than the times employed by the ϵ-constraint method. In particular, the time improvement given by the selected two-phase method, with respect to the PCTSP version of the ϵ-constraint method is, on average, 22% on randomly generated instances and 13% on instances from TSPLIB. If we analyze the time difference with respect to the OP version of the ϵ-constraint method, we notice that the selected two-phase method gives a 60% improvement on randomly generated instances and 40% on instances from TSPLIB. The advantage of the two-phase method is dramatic on small to medium sized instances with a tight range of profits, but is less apparent on large instances with a wide range of profits, namely the hardest instances. It is worth to point out that many of the latter instances are unsolvable by both methods with the available computing resources, and that in such instances the cardinality of the efficient set is so high that the practical relevance of its exact description might be questionable.

An important point is the number of solved subproblems. In the BE algorithm, every optimal solution of a $P(\lambda, \phi_2, \gamma_1)$ problem corresponds to an efficient point, whereas, in the ϵ-constraint algorithm, an optimal solution of a $P_1(\epsilon)$ or $P_2(\epsilon)$ problem is not guaranteed to correspond to an efficient point. Thus, it may happen that the ϵ-constraint method, in any one of its versions, solves a larger number of subproblems than the BE algorithm. Furthermore, on the same instance, the number of non-efficient points that are generated may greatly vary if either problems $P_1(\epsilon)$ or problems $P_2(\epsilon)$ are solved. On a few of the instances we have tested, the running times of the two implementations have differed by a factor of 30. Though in the TSPP case the choice of $P_1(\epsilon)$, corresponding to a PCTSP, turns out to be convenient ([2]), in general there is no apparent rule to decide a priori which is the most convenient problem to solve. On the other hand, if we analyze the performances of the BE algorithm and the two-phase BE algorithm we can notice a stable behavior in the time trend.

We therefore conclude that the BE algorithm and its improvements are more stable in generic cases, especially where single objective subproblems cannot be reduced to well known problems, for which specific resolution techniques are available, or when it is not apparent which objective function is better to constrain in a single objective subproblem.

6. Conclusions

By elaborating some ideas proposed in the literature for specific problems, in this paper we discuss a general two-phase method suitable for hard BOCO problems. We compared our method with the ϵ-constraint method on the Traveling Salesman Problem with Profits, where the incurred costs and the collected profits are treated as conflicting objectives. On a large set of instances we observe that the approach suggested in this paper is more competitive and more stable than the ϵ-constraint method.

The proposed procedure can be used with little modifications to develop approximation schemes able to detect subsets of exact efficient points that approximate the Pareto optima within a specified error on both objectives, in the spirit of [11]. We have developed this subject in a companion paper ([8]).

References

[1] Y. P. Aneja, K. P. K. Nair, Bicriteria Transportation Problem. Management Science. 25(1979)73-78 .

[2] Bérubé J., M. Gendreau, J. Potvin, An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits. Eur. J. Oper. Res. 194(2009) 39-50 .

[3] S. Coene, C. Filippi, F. C. R. Spieksma, E. Stevanato, Balancing profits and costs on trees. Networks. 61(2013) 200-211 .

[4] J. L. Cohon, Multiobjective Programming and Planning (Academic Press, New York, 1978).

[5] M. Ehrgott, X. Gandibleux, in Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys, M. G. Ehrgott, X. Gandibleux, Eds. (Kluwer Academic, Boston, 2002), pp. 369-444, vol. 52.

[6] M. Ehrgott, X. Gandibleux, A survey and annotated bibliography of multi-objective Combinatorial Optimization. OR Specktrum. 22(2000) 425-460 .

[7] D. Feillet, P. Dejax, M. Gendreau, Traveling Salesman Problems with Profits. Transportation Science. 39(2005) 188-205 .

[8] C. Filippi, E. Stevanato, Approximation schemes for bi-objective combinatorial optimization and their application to the TSP with profits. Computers & Operations Research.(2013).

Table 1

			Constrained 1st Phase	Unconstrained 1st Phase
Profit Class	$\|V\|$	$\|\mathcal{SE}\|$	$Time$	$Time$
I	15	12	0.3	**0.1**
	25	25	3	**1.2**
	35	11	17	**3**
	50	16	46	**14**
II	15	14	1.5	**0.4**
	25	19	8	**1.2**
	35	36	153	**18**
	50	67	989	**120**
III	15	23	1	**0.2**
	25	52	26	**8**
	35	46	193	**41**
	50	112	2953	**1822**

Computation of the supported set on randomly generated instances.

Table 2

			BE	FBE	$FBE+LK$	$2P$	$2P+FUB$
Profit Class	$\|V\|$	$\|\mathcal{E}\|$	$Time$	$Time$	$Time$	$Time$	$Time$
I	15	18	0.9	0.5	0.5	0.1	**0.1**
	25	37	4	3.1	3.2	3.1	**3**
	35	48	30	24	25	21	**20.8**
	50	69	267	231	235	**167**	170
II	15	37	2	1	0.9	**0.7**	0.8
	25	72	21	16	17	16	**16**
	35	142	210	190	192	182	**178**
	50	207	2612	2247	2284	1556	**1540**
III	15	34	2	1.2	1.2	1.2	**1.2**
	25	105	37	34	34	32	**28**
	35	140	308	290	291	**279**	290
	50	235	4502	4325	4327	3651	**3646**

Different versions of the BE algorithm on randomly generated instances.

Table 3

		PCTSP ϵ-constraint		OP ϵ-constraint		$2P+FUB$	
Profit Class	$\|V\|$	$Time$	N	$Time$	N	$Time$	$\|\mathcal{E}\|$
I	15	0.8	18	4	48	**0.1**	18
	25	5.5	37	18	96	**3**	37
	35	25	48	102	180	**20.8**	48
	50	250	70	1113	292	**170**	69
II	15	1	38	5	60	**0.8**	37
	25	18	74	40	152	**16**	72
	35	200	142	301	290	**178**	142
	50	1720	209	2659	420	**1540**	207
III	15	1.5	37	3	57	**1.2**	34
	25	33	109	43	185	**28**	105
	35	292	144	323	240	**290**	140
	50	3754	236	**3365**	423	3646	235

The ϵ-constraint method and the two-phase implementation of the BE algorithm, on randomly generated instances.

Table 4

			Constrained 1st Phase	Unconstrained 1st Phase
Instance	$\|V\|$	$\|\mathcal{SE}\|$	$Time$	$Time$
burma14	14	5	0.5	**0**
ulysses16	16	6	0.6	**0**
ulysses22	22	8	1.3	**0.3**
att48	48	10	17	**6**
eil51	51	16	25	**12**
berlin52	52	18	44	**21**
st70	70	27	460	**211**
eil76	76	25	3121	**1785**
pr76	76	29	15775	**3643**

Computation of the supported set for TSPLIB instances with profits of type A.

Table 5

			BE	FBE	$FBE+LK$	$2P$	$2P+FUB$
Instance	$\|V\|$	$\|\mathcal{E}\|$	$Time$	$Time$	$Time$	$Time$	$Time$
burma14	14	13	1	0.7	0.7	**0.1**	**0.1**
ulysses16	16	15	1	0.7	0.68	0.2	**0.1**
ulysses22	22	21	2	1.5	1.6	**1.6**	**1.6**
att48	48	47	65	58	60	**51**	54
eil51	51	50	45	39	39	34	**25**
berlin52	52	51	68	62	61	**50**	52
st70	70	69	730	668	675	476	**462**
eil76	76	75	4696	4410	4486	4112	**4099**
pr76	76	51	t.l.e.	t.l.e.	t.l.e.	23612*	23522*

Different versions of the BE algorithm on TSPLIB instances with profits of type A.

Table 6

			BE	FBE	$FBE+LK$	$2P$	$2P+FUB$
Instance	$\|V\|$	$\|\mathcal{E}\|$	$Time$	$Time$	$Time$	$Time$	$Time$
burma14	14	59	4.1	3.4	3.4	1.1	**1.0**
ulysses16	16	102	6.2	5.7	5.7	**1.3**	**1.3**
ulysses22	22	130	25	19	20	18	**14**
att48	48	435	3476	2984	2997	1511	**1497**
eil51	51	225	1951	1765	1798	1510	**1501**
berlin52	52	406	2980	2620	2672	1442	**1386**
st70	70	503	20445	19997	18995	15060	**14693**
eil76	76	386	41410	40755	40845	38920	**38116**
pr76	76	25	t.l.e.	t.l.e.	t.l.e.	6007*	5992*
rat99	99	662	56399	45005	46721	43855	**43673**

Different versions of the BE algorithm on TSPLIB instances with profits of type B.

[9] X. Gandibleux, H. Morita, N. Katoh, Use of a genetic heritage for solving the assignment problem with two objectives. Lecture Notes in Computer Sciences. 2632(2003) 43-57 .

[10] A. Goldberg, R. Tarjan, A new approach to the Maximum Flow Problem. Journal of the Association for Computing Machinery. 35(1988) 921-940 .

[11] H. W. Hamacher, C. R. Pedersen, S. Ruzika, Finding representative systems for discrete bicriterion optimization problems. Oper. Res. Lett. 35(2007) 336-344 .

[12] K. Helsgaun, An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur. J. Oper. Res. 126(2000) 106-130 .

[13] M. Köksalan, Multiobjective Combinatorial Optimization: Some Approaches. Journal of Multi-Criteria Decision Analysis. 15(2009) 69-78 .

[14] J. R. Ledesma, J. J. Salazar, The bi-objective travelling purchaser problem. Eur. J. Oper. Res. 160(2005) 599-613 .

Table 7

			BE	FBE	$FBE+LK$	$2P$	$2P+FUB$
Instance	$\|V\|$	$\|\mathcal{E}\|$	$Time$	$Time$	$Time$	$Time$	$Time$
burma14	14	70	3.8	3.5	3.5	1.2	**1.1**
ulysses16	16	92	6.1	5.5	5.6	2.1	**2.1**
ulysses22	22	128	40	34.9	35.7	27	**26.8**
att48	48	438	7336	6461	6451	4112	**4048**
eil51	51	267	5655	5312	5365	5001	**4892**
berlin52	52	439	6614	5441	5742	4365	**4361**
st70	70	452	28769	21468	22576	14411	**14238**
eil76	76	383	8554	7987	8012	7155	**6924**
pr76	76	31	t.l.e.	t.l.e.	t.l.e.	10927*	12314*

Different versions of the BE algorithm on TSPLIB instances with profits of type C.

Table 8

		PCTSP ϵ-constr.		OP ϵ-constr.		$2P+FUB$	
Instance	$\|V\|$	$Time$	N	$Time$	N	$Time$	$\|\mathcal{E}\|$
burma14	14	0.3	14	1	23	**0.1**	14
ulysses16	16	0.4	16	1	30	**0.1**	16
ulysses22	22	2	22	8	76	**1.6**	22
att48	48	**53**	48	274	196	54	48
eil51	51	33	51	943	308	**25**	51
berlin52	52	54	52	1012	509	**52**	52
st70	70	600	70	2428	260	**462**	70
eil76	76	**3680**	76	13744	312	4099	76
pr76	76	21587*	40	25423*	42	23522*	51
rat99	99	365	99	1922	233	**349**	99
KroA100	100	9216	100	15743	99	**9099**	100
KroB100	100	19665	100	19322	111	**18514**	100
KroC100	100	12132	100	14232	117	**11563**	100
KroD100	100	3654	100	4034	121	**3402**	100
KroE100	100	4943	100	5254	111	**4304**	100
rd100	100	3931	100	4521	109	**3744**	100
eil101	101	17433	101	15466	100	**14362**	101
lin105	105	38492	105	44638	114	**36711**	105
pr107	107	3376	106	3869	136	**3341**	107
pr124	124	43655	124	44571	162	**42021**	124
bier127	127	4123	127	4539	129	**3764**	127
ch130	130	8195	130	8731	156	**7793**	130
pr136	136	47509*	80	52140*	78	41435*	79
gr137	137	31585	137	40022	149	**31021**	137
pr144	144	171098	144	199421	193	**164915**	144
KroA150	150	15052*	19	13978*	16	13243*	17
KroB150	150	76982*	72	85312*	73	75677*	74
ch150	150	17599*	30	19764*	37	18553*	38
pr152	152	19223*	5	21232*	3	16631*	4
u159	159	5123*	12	5522*	8	4743*	11

The ϵ-constraint method and the two-phase implementation of the BE algorithm on TSPLIB instances with profits of type A.

Table 9

		PCTSP ϵ-constr.		OP ϵ-constr.		$2P + FUB$	
Instance	$\|V\|$	$Time$	N	$Time$	N	$Time$	$\|\mathcal{E}\|$
burma14	14	2	59	5	78	**1.0**	59
ulysses16	16	4.8	102	8	186	**1.3**	102
ulysses22	22	15	130	26	159	**14**	130
att48	48	**1425**	438	1844	487	1497	435
eil51	51	1567	269	**943**	308	1501	225
berlin52	52	**1369**	411	3474	561	1386	406
st70	70	17500	643	21321	735	**14693**	503
eil76	76	45283	538	47287	566	**38116**	386
pr76	76	6233*	24	7342*	21	5872*	25
rat99	99	45243	779	59966	799	**43573**	662
KroA100	100	82897*	156	96758*	157	95673*	256
KroB100	100	119443*	441	131865*	466	780952*	911
KroC100	100	13968*	82	16432*	80	8813*	82
KroD100	100	85331*	778	89951*	767	72353*	790
KroE100	100	71111*	821	78374*	800	60101*	815
rd100	100	101443*	870	144397*	869	92158*	868
eil101	101	143266*	332	155683*	321	123454*	335

The ϵ-constraint method and the two-phase implementation of the BE algorithm on TSPLIB instances with profits of type B.

Table 10

		PCTSP ϵ-constr.		OP ϵ-constr.		$2P + FUB$	
Instance	$\|V\|$	$Time$	N	$Time$	N	$Time$	$\|\mathcal{E}\|$
burma14	14	3	70	2	78	**1.1**	70
ulysses16	16	3.86	92	5	153	**2.1**	92
ulysses22	22	**18**	128	35	198	26.8	128
att48	48	3733	440	**3298**	518	4048	438
eil51	51	6090	299	5878	287	**4892**	267
berlin52	52	3990	446	**3984**	561	4361	439
st70	70	14538	546	34451	558	**14238**	452
eil76	76	7411	468	10234	465	**6924**	383
pr76	76	11243*	29	18538*	26	12314*	31
rat99	99	4236*	13	5476*	13	2954*	15
KroA100	100	36978*	101	65716*	99	100001*	416
KroB100	100	33131*	35	58732*	37	61011*	160
KroC100	100	25537*	123	34521*	122	19847*	121
KroD100	100	68768*	211	67556*	215	50122*	236
KroE100	100	97643*	176	100254*	184	83322*	184
rd100	100	89432*	336	94543*	304	83236*	354
eil101	101	84335*	383	98231*	479	77361*	387

The ϵ-constraint method and the two-phase implementation of the BE algorithm on TSPLIB instances with profits of type C.

[15] M. Müller-Hannemann, K. Weihe, On the cardinality of the Pareto set in bicriteria shortest path problems. Annals of Operations Research. 147(2006) 269-286 .

[16] G. Mavrotas, Effective implementation of the epsilon-constraint method in Multi-Objective Mathematical Programming problems. Applied Mathematics and Computation. 213(2009) 455-465 .

[17] A. Przybylski, X. Gandibleux, M. Ehrgott, A two-phase method for multi-objective integer programming and its application to the assignment problem with three objectives. Discrete Optimization. 7(2010) 149-165 .

[18] A. Przybylski, X. Gandibleux, M. Ehrgott, Two phase algorithms for the bi-objective assignment problem. Eur. J. Oper. Res. 185(2008) 509-533 .

[19] A. Przybylski, X. Gandibleux, M. Ehrgott, Two-phase algorithms for Bi-objective Assignment Problem. Eur. J. Oper. Res. 185(2008) 509-533 .

[20] S. Raghavan, M. O. Ball, V. S. Trichur, Bicriteria product design optimization: an efficient solution procedure using AND/OR trees. Naval Research Logistics. 49(2002) 574-592 .

[21] A. Raith, M. Ehrgott, A two-phase algorithm for the biobjective Integer Minimum Cost Flow Problem. Computers & Operation Research. 36(2009) 1945-1954 .

[22] G. Reinelt, A traveling salesman problem library. ORSA Journal of Computing. 3(1991) 376-384 .

[23] D. Tuyttens, J. Teghem, P. Fortemps, K. Van Nieuwenhuyse, Performance of the MOSA Method for the Bicriteria Assignment Problem. Journal of Heuristic. 6(2000) 295-310 .

Received 21-8-2012; accepted 29-4-2013