CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

An Exact ε-constraint Method for Bi-objective Combinatorial Optimization Problems - Application to the Traveling Salesman Problem with Profits

Jean-François Bérubé
Michel Gendreau
Jean-Yves Potvin

February 2007

CIRRELT-2007-05

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince, local 2642
Québec (Québec)
Canada G1K 7P4
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

An Exact ε-constraint Method for Bi-objective Combinatorial Optimization Problems – Application to the Traveling Salesman Problem with Profits

Jean-François Bérubé[1,*], Michel Gendreau[1], Jean-Yves Potvin[1]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 and Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7

Abstract. This paper describes an exact ε-constraint method for bi-objective combinatorial optimization problems with integer objective values. This method tackles multi-objective optimization problems by solving a series of single objective subproblems, where all but one objective are transformed into constraints. We show in this paper that the Pareto front of bi-objective problems can be efficiently generated with the ε-constraint method. Furthermore, we describe heuristics based on information gathered from previous subproblems that significantly speed up the method. This approach is used to find the exact Pareto front of the Traveling Salesman Problem with Profits, a variant of the Traveling Salesman Problem in which a profit or prize value is associated with each vertex. The goal here is to visit a subset of vertices while addressing two conflicting objectives: maximize the collected prize and minimize the travel costs. We report the first exact results for this problem on instances derived from classical vehicle routing and traveling salesman problem instances. Results on approximations of the Pareto front obtained from a variant of our exact algorithm are also reported.

Keywords. Bi-objective combinatorial optimization, ε-constraint problem, Pareto front, branch-and-cut, traveling salesman problem with profits.

Acknowledgements. Financial support for this work was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds québécois de la recherche sur la nature et les technologies (FQRNT). This support is gratefully acknowledged.

---

* Corresponding author: Jean-Francois.Berube@cirrelt.ca

This document is also published as Publication #1295 by the Department of Computer Science and Operations Research (DIRO) of the Université de Montréal.

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2007

Introduction

Decision making issues can rarely rely on a single well defined criterion. Although the multiple facets of a decision process can be aggregated into a single objective function, this simplification involves arbitrary rules that can hardly capture adequately the complexity of real world decision issues. Thereby, the interest for multi-criteria decision making has continually grown during the last decades, as attested by the number of books and surveys on the topic (see [6, 10, 22, 27], among others). It comes as no surprise if more and more publications address combinatorial issues given that many real world applications involve discrete decisions or events. The reader is referred to [9] for a review of the literature on multi-objective combinatorial optimization (MOCO) problems.

This paper addresses a special case of MOCO problems, namely bi-objective combinatorial optimization (BOCO) problems with integer objective values. BOCO problems are often considered independently of MOCO problems because of their particular nature: going from many to two objectives corresponds to a significant simplification of the problem (*"Three is more than two plus one."* [9]). General BOCO problems are formulated as:

$$\min\ f(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x})) \qquad \text{subject to: } \vec{x} \in \mathcal{X} \tag{1}$$

where $\mathcal{X}$ is the set of feasible solutions, or *solution space*. We denote each evaluation vector $f(\vec{x})$ as $\vec{z}$ and $(\vec{z})_i$ stands for the value of the ith objective. To simplify the notation, we write z_i instead of $(\vec{z})_i$ up to Theorem 3. From there, $(\vec{z})_i$ is used to avoid any ambiguity.

The *objective space* is defined by $\mathcal{Z} = \{\vec{z} = (z_1, z_2) : z_i = f_i(\vec{x}),\ \forall \vec{x} \in \mathcal{X},\ i = 1, 2\}$. Since no solution optimizes simultaneously both objectives, one will search for an acceptable trade-off instead of an optimal solution. This compromise must be such that no strictly better solution exists, even though some solutions might be considered as equivalent. This involves a partial order of the objective space, defined by a *dominance relation*. The latter is used to characterize *Pareto efficiency*, a concept that replaces the optimal solution of single objective optimization problems.

Definition 1 (Dominance relation). *Let $\vec{z}$ and $\vec{z}\,' \in \mathcal{Z}$. We say that $\vec{z}$ dominates $\vec{z}\,'$ ($\vec{z} \succ \vec{z}\,'$) if and only if $z_1 \leq z'_1$ and $z_2 \leq z'_2$ where at least one inequality is strict.*

Definition 2 (Pareto efficiency). *A solution $\vec{x} \in \mathcal{X}$ is (Pareto) efficient in $\mathcal{X}$, if and only if $\nexists \vec{x}\,' \in \mathcal{X}$ such that $f(\vec{x}\,') \succ f(\vec{x})$.*

Definition 3 (Efficient set). *The efficient set $\mathcal{E} = \{\vec{x} \in \mathcal{X} : \vec{x}$ is Pareto efficient in $\mathcal{X}\}$.*

Definition 4 (Pareto front). *The Pareto front $\mathcal{F} = \{f(\vec{x}) : \vec{x} \in \mathcal{E}\}$.*

The *Efficient set* ($\mathcal{E}$) and *Pareto front* ($\mathcal{F}$) contain all the Pareto efficient solutions and all the non-dominated points in the objective space, respectively. Since the efficient set is defined on the solution space while the Pareto front is defined on the objective space, the cardinality of $\mathcal{E}$ is always greater than or equal to the cardinality of $\mathcal{F}$. That is, there might be many feasible solutions that correspond to the same point in

the objective space. Usually the aim of multi-objective optimization is to find one solution for each point of $\mathcal{F}$ or to approximate the Pareto front with an efficient set of solutions.

Among exact methods to find the Pareto front of MOCO problems, weighted sum scalarization is the most popular according to [9]. This method solves different single objective subproblems generated by a linear scalarization of the objectives. By varying the weights of this linear function, all supported[1] non-dominated points can be found. It is worth noting that the subproblems are as easy to solve as the corresponding mono-criterion problems. On the other hand, linear scalarization cannot find unsupported points and is therefore ill-suited for non-convex objective spaces such as those associated with MOCO problems. This drawback can be overcome with the Two-Phase Method [29] that finds all supported points of $\mathcal{F}$ through a weighted sum scalarization in the first phase, while non-supported points are found during the second phase with problem specific methods. Most algorithms that find the exact Pareto front of MOCO problems are variants of the Two-Phase Method [9], although other parametric approaches based on weighted scalarizations can find the exact Pareto front of BOCO problems [19, 23, 26].

Besides weighting sum algorithms, the ϵ-constraint method [6, 22] is the best known approach for solving MOCO problems, according to [9]. This method generates single objective subproblems, called ϵ-constraint problems, by transforming all but one objectives into constraints. The upper bounds of these constraints are given by the ϵ-vector and, by varying it, the exact Pareto front can theoretically be generated. In practice, because of the high number of subproblems and the difficulty to establish an efficient variation scheme for the ϵ-vector, this approach has mostly been integrated within heuristic and interactive schemes. It can however generate the exact Pareto front in particular situations, such as BOCO problems, as we will see later.

This paper focuses on BOCO problems for which no polynomial time algorithm exist for solving the corresponding single objective problems, but where the latter can still be efficiently solved through branch-and-cut. Many problems share these characteristics, including the Bi-Objective Covering Tour problem [16] and bi-objective variants of the Traveling Salesman Problem (TSP), such as the Bi-Objective Traveling Purchaser Problem [25] and the Traveling Salesman Problem with Profits (TSPP) [11]. For these problems, ϵ-constraint methods are particularly attractive because the addition of new constraints through a branch-and-cut procedure is quite natural.

The first contribution of this paper is to show the correctness of an efficient variant of the ϵ-constraint method for BOCO problems, where exactly one ϵ-constraint problem is solved for each point on the Pareto front. The second contribution is the introduction of heuristic improvements based on the exploitation of information gathered from previous problems that provides significant speed-ups. The proposed method is then used to solve instances of the TSPP, a variation of the TSP in which a profit or prize value is associated with each vertex. We report the first exact Pareto fronts for TSPP instances obtained from classical VRP and TSP instances available in the TSPLIB [24]. The paper is organized as follows. Our general problem-solving approach is presented in Section 1. Then, Section 2 introduces the improvement heuristics. Section 3 describes the TSPP and explains how our general algorithm can be adapted to solve it. Finally, Section 4 reports computational results on several TSPP instances.

[1] The supported points are those found on the convex envelope of the objective space.

1 Exact ϵ-constraint method for BOCO problems

The ϵ-constraint method has been developed for general multi-objective problems. It solves ϵ-constraint problems $P_k(\epsilon)$ obtained by transforming one of the objectives into a constraint. For the bi-objective case, the $P_i(\epsilon_j)$ problems are:

$$\min f_i(\vec{x}) \tag{2}$$

$$\text{subject to:} \quad \vec{x} \in \mathcal{X}, \tag{3}$$

$$f_j(\vec{x}) \leq \epsilon_j, \quad i,j = 1,2 \;\; i \neq j. \tag{4}$$

Theorem 1. *$\vec{x}^*$ is an efficient solution of a BOCO problem if and only if $\vec{x}^*$ solves $P_i(\epsilon_j)$ for $i = 1, 2$.*

Theorem 2. *If $\vec{x}^*$ solves $P_i(\epsilon_j)$ for some i and if this solution is unique, then $\vec{x}^*$ is an efficient solution of a BOCO problem.*

Theorems 1 and 2 are proved for general multi-objective problems (see [6, 22]) and are therefore valid for BOCO problems. These theorems mean that efficient solutions can always be found by solving ϵ-constraint problems, as long as ϵ_j is such that $P_i(\epsilon_j)$ is feasible. Moreover, Theorem 1 indicates that for any efficient solution $\vec{x}^*$, one can find an ϵ_j such that $\vec{x}^*$ solves $P_i(\epsilon_j)$ for $i = 1, 2$. In other words, the exact Pareto front can be found by solving ϵ-constraint problems, as long as we know how to modify ϵ_j to generate at least one solution for every point of $\mathcal{F}$. This issue has recently been addressed for the general multi-objective case in [21], but it remains an important drawback of the ϵ-constraint method. However, the particularities of BOCO problems yield to a simple variation scheme for ϵ that can be numerically implemented. The idea is to construct a sequence of $P_i(\epsilon_j)$ problems based on a progressive reduction of ϵ_j. Let $\vec{z}^I = (z_1^I, z_2^I)$ with

$$z_1^I = \min_{\vec{z} \in \mathcal{Z}} z_1 \quad \text{and} \quad z_2^I = \min_{\vec{z} \in \mathcal{Z}} z_2, \tag{5}$$

be the ideal point and let $\vec{z}^N = (z_1^N, z_2^N)$ with

$$z_1^N = \min_{\vec{z} \in \mathcal{Z}} \{z_1 : z_2 = z_2^I\} \quad \text{and} \quad z_2^N = \min_{\vec{z} \in \mathcal{Z}} \{z_2 : z_1 = z_1^I\} \tag{6}$$

be the Nadir point that defines lower and upper bounds on the value of efficient solutions, respectively. Algorithm 1 finds $\mathcal{F}$ through a sequence of ϵ-constraint problems. Throughout the algorithm, ϵ_j is decreased by a constant value Δ (currently set to 1). As explained later, Δ may sometimes be larger to strengthen the ϵ-constraint. Note that a similar approach is used without proof in [16] for the Bi-Objective Covering Tour problem.

This strategy is somehow related to ranking methods, another exact problem-solving scheme for bi-objective problems. The idea of ranking methods is to start from a feasible solution $\vec{x}$ such that $f_1(\vec{x}) = z_1^I$ (or $f_2(\vec{x}) = z_2^I$) and to find the second best, third best, ..., feasible solutions based on the first (or second) objective, until the Nadir point in reached. Among the resulting solutions, there is a set of efficient solutions that represent all points on the Pareto front. This approach has been introduced in the early 80's to solve

Algorithm 1 : Exact Pareto front of BOCO problems with integer objective values.

1. Set $i = 1,\ j = 2$ or $i = 2,\ j = 1$.
2. Compute the ideal and Nadir points.
3. Set $\mathcal{F} = \{(z_i^I, z_j^N)\}$ and $\epsilon_j = z_j^N - \Delta$ ($\Delta = 1$).
4. While $\epsilon_j \geq z_j^I$, do:
   (a) Solve $P_i(\epsilon_j)$ through branch-and-cut and add the optimal solution value $(z_i^*, z_j^*)$to $\mathcal{F}$.
   (b) Set $\epsilon_j = z_j^* - \Delta$.
5. Remove dominated points from $\mathcal{F}$ if required (as explained later, some dominated points might be found throughout this procedure).

the bi-objective shortest path problem [7]. It relies on polynomial time K-best algorithms, which have been developed for many problems such as the shortest path and minimum spanning tree problems. To prove the correctness of Algorithm 1, we first state two lemmas similar to those in [7]. However, the latter are specific to the shortest path problem while our discussion takes place in the context of general BOCO problems.

Lemma 1. *$(z_1^I, z_2^N) \in \mathcal{F}$ and $(z_1^N, z_2^I) \in \mathcal{F}$.*

Proof. Suppose that $(z_1^I, z_2^N) \notin \mathcal{F}$. Then, $\exists (z_1, z_2) \in \mathcal{Z} : (z_1, z_2) \succ (z_1^I, z_2^N)$. Thus according to definition 1, either:

1) $z_1 < z_1^I$ and $z_2 < z_2^N$ or

2) $z_1 < z_1^I$ and $z_2 = z_2^N$ or

3) $z_1 = z_1^I$ and $z_2 < z_2^N$.

Since 1) and 2) contradict the definition of an ideal point and because 3) contradicts the definition of a Nadir point, then $(z_1^I, z_2^N) \in \mathcal{F}$. The proof that $(z_1^N, z_2^I) \in \mathcal{F}$ is similar. □

Lemma 2. *$\forall (z_1, z_2) \in \mathcal{Z}$, if $(z_1, z_2) \in \mathcal{F}$, then $z_1^I \leq z_1 \leq z_1^N$ and $z_2^I \leq z_2 \leq z_2^N$.*

Proof. By Lemma 1, $(z_1^I, z_2^N) \in \mathcal{F}$, thus it is non-dominated. Since $z_1^I = \min_{\bar{z} \in \mathcal{Z}} z_1$, $z_1 \geq z_1^I$, $\forall (z_1, z_2) \in \mathcal{F}$. Also, if $z_2 > z_2^N$, $(z_1^I, z_2^N) \succ (z_1, z_2)$ and $(z_1, z_2) \notin \mathcal{F}$. Hence, $z_1 \geq z_1^I$ and $z_2 \leq z_2^N$, $\forall (z_1, z_2) \in \mathcal{F}$. The proofs for $z_2 \geq z_2^I$ and $z_1 \leq z_1^N$ are similar. □

Lemma 2 defines a region of the objective space that contains the Pareto front. Let us define the set

$$\mathcal{Z}^+ = \{(z_1, z_2) \in \mathcal{Z} : z_1^I \leq z_1 \leq z_1^N \text{ and } z_2^I \leq z_2 \leq z_2^N\} \tag{7}$$

which is depicted in Figure 1 by the square region formed by the ideal and Nadir points. This region can be used to characterize the interval of possible values for ϵ_j, i.e. $[z_j^I, z_j^N]$. Let ϵ_j^s be the sth value on this interval and let us assume that $\mathcal{Z}^+$ is made of subsets

$$Z^+_{\epsilon_j^s} = \{\vec{z} \in \mathcal{Z}^+ : z_i = f_i(\vec{x}^*) \text{ where } \vec{x}^* \text{ is a solution of } P_i(\epsilon_j^s)\}. \quad (8)$$

Figure 1 (a) depicts a typical point $\vec{z}^*$ that minimizes the second objective among the points of $\mathcal{Z}^+_{\epsilon_2^s}$ and it shows the preferred and dominated regions according to this point. If $\vec{z}^* \in \mathcal{F}$, the other points of the Pareto front must be in the two unidentified regions of $\mathcal{Z}^+$. The correctness of Algorithm 1 is shown by the following theorem.

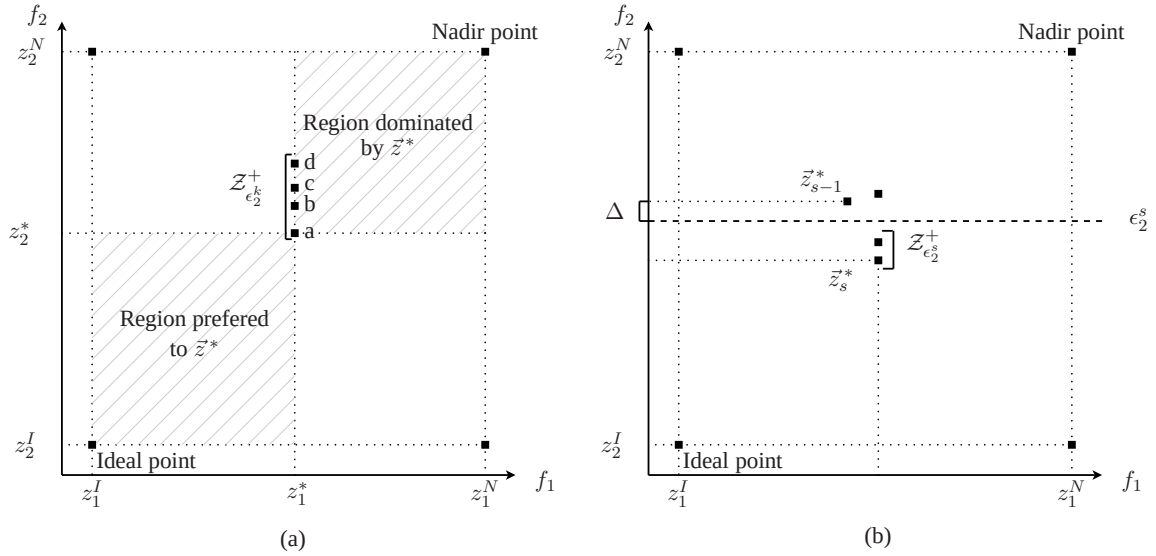

Figure 1: (a) Illustration of the dominance relation among elements of $\mathcal{Z}^+$. (b) Illustration of two consecutive points in the sequence defined by Theorem 3.

Theorem 3. *A sequence of ϵ-constraint problems $P_i(\epsilon_j)$ defined by $\epsilon_j^1, ..., \epsilon_j^s, ..., \epsilon_j^S$ where:*

(a) $\epsilon_j^1 = z_j^N$, $\epsilon_j^S = z_j^I$,

(b) $\epsilon_j^s = (\vec{z}^*_{s-1})_j - \Delta$, *with* $\vec{z}^*_{s-1}$ *the value of a solution to* $P_i(\epsilon_j^{s-1})$ *and* $\Delta = 1$,

generates one feasible solution for each point of the Pareto front.

Proof. Let $\vec{z}^*_1, ..., \vec{z}^*_s, ..., \vec{z}^*_S$ be the sequence of solutions corresponding to the sequence of ϵ-constraint problems defined by (a) and (b). Let us show that if $\vec{z} \in \mathcal{Z} \setminus \{\vec{z}^*_1, ..., \vec{z}^*_s, ..., \vec{z}^*_S\}$, then $\vec{z} \notin \mathcal{F}$. Assume that there is a solution $\vec{z}\,' \in \mathcal{Z} \setminus \{\vec{z}^*_1, ..., \vec{z}^*_s, ..., \vec{z}^*_{|\mathcal{F}|}\}$ such that $\vec{z}\,' \in \mathcal{F}$. By Lemma 2, $z_i^I \leq z'_i \leq z_i^N$. Then, either:

1) $z'_i = (z^*_s)_i$ (for a given s, $s = 1, ..., S$), or

2) $(\vec{z}^*_{s-1})_i < z'_i < (\vec{z}^*_s)_i$ and $(\vec{z}^*_{s-1})_j < z'_j \leq (\vec{z}^*_s)_j$ (for a given s, $s = 1, ..., S$).

In case 1), z'_j must be lower than $(\vec{z}^*_s)_j$ for $\vec{z}'$ to be efficient. But since Δ equals 1 and since the objective values are integers, ϵ_j will eventually reach a value for which the optimum of the corresponding ϵ_j-constraint problem is $\vec{z}'$, that is $\vec{z}' \in \{\vec{z}^*_{s+1}, ..., \vec{z}^*_S\}$, which contradicts the hypothesis. Case 2) is impossible because $\vec{z}^*_s$ is the optimal value of $P_i(\epsilon_j^{s-1} - \Delta)$, with $\Delta = 1$ and integer objectives.

□

Handling the dominated points. Because there might exist many solutions to $P_i(\epsilon_j^s)$ with different values for objective j (i.e. $|Z^+_{\epsilon_j^s}| > 1$), some dominated points might be generated by the sequence of ϵ-constraint problems defined by Theorem 3. For example, in Figure 1(a), the points b, c and d are dominated by point a. Nevertheless, since all non-dominated points will be found, one can simply exclude all the non-efficient solutions to obtain the exact Pareto front, as it is done with the K-best solutions obtained from ranking methods. Another possibility is to solve both $P_1(\epsilon_2^s)$ and $P_2(\epsilon_1^s)$ (see Theorem 1). This can be done implicitly by modifying the branch-and-cut algorithm used to solve the ϵ-constraint problems. Let $\vec{z}^*_s$ be the optimal value for $P_i(\epsilon_j^s)$. Then, the lower bound of all pending nodes in the branching tree are greater than or equal to $(\vec{z}^*_s)_i$, and other optimal solutions might be found by processing the pending nodes with a lower bound equal to $(\vec{z}^*_s)_i$. Moreover, since a feasible solution such that $(\vec{z}^*_s)_j \leq \epsilon_j^s$ is known, ϵ_j^s can be decreased to $(\vec{z}^*_s)_j - \Delta$. Doing so until no more feasible solution exists will lead to a unique optimal solution for the strengthened ϵ-constraint problem, which satisfies Theorem 2. This strategy reduces the number of subproblems to the exact number of points on the Pareto front, but those problems might be harder to solve.

The efficient set. A similar modification of the branch-and-cut algorithm produces the efficient set in addition to the Pareto front. The idea is to fathom only the nodes with a lower bound strictly greater than the value of the best known feasible solution. The value of ϵ_j^s should be updated to the maximum of $(\vec{z}^*)_j$ and ϵ_j, for each optimal value $\vec{z}^*_s$ found. Moreover, inequalities that cuts all known optimal solutions must be added.

Approximation. One should also observe that the algorithm can be modified to generate approximations of the Pareto front while reducing the computation time. Since the branch-and-cut procedure runs until the gap between the best known feasible solution value (upper bound) and the best linear relaxation value (lower bound), a tolerance on this gap will produce approximate solutions for the ϵ-constraint problems and thus generate an approximate Pareto front. As shown in Section 4, this strategy significantly reduces the computation time while providing good approximations of the Pareto front.

Strengthening the ϵ-constraint. Until now, we have considered that the constant Δ equals 1, but in some cases, it might be possible to use a larger value, which obviously strengthens the ϵ-constraint. We show that Δ can be set to the greatest common divisor among the possible values of objective j (Ω_j), assuming integer objective values.

Proposition 1. $\forall \vec{z}_1, \vec{z}_2 \in \mathcal{Z}$ *such that* $(\vec{z}_1)_j \neq (\vec{z}_2)_j$, $|(\vec{z}_1)_j - (\vec{z}_2)_j| \geq \Omega_j$

Proof. Suppose there exist some points $\vec{z}_1$ and $\vec{z}_2 \in \mathcal{Z}$ such that $|(\vec{z}_2)_j - (\vec{z}_2)_j| < \Omega_j$, then

$$\frac{|(\vec{z}_1)_j - (\vec{z}_2)_j|}{\Omega_j} < 1 \tag{9}$$

Since objective values are positive integers, the left-hand side of equation (9) must be null. Therefore, $(\vec{z}_1)_j = (\vec{z}_2)_j$ which contradicts the hypothesis. □

2 Improvement Heuristics

The sequence of ϵ-constraint problems is defined by a progressive reduction of ϵ_j that leads to a progressive increase in objective i. This suggests that the structure of consecutive subproblems might be similar. In the following, two classes of heuristics are proposed to take advantage of these similarities in order to improve the branch-and-cut algorithm when solving $P_i(\epsilon_j)$. Both heuristics exploit information gathered from previous subproblems to solve future subproblems faster. The first heuristic exploits knowledge of the polytope, while the second one improves the quality of the initial solution. Later on, we will explain how those heuristics can be applied in the case of the TSPP.

Exploiting knowledge of the polytope. The general principle underlying cutting-plane algorithms is to reduce the size of the relaxed solution space by adding valid cuts. Due to structural similarities between two consecutive ϵ-constraint problems, and since separation of violated inequalities is often hard, it is quite natural to keep some constraints from one $P_i(\epsilon_j)$ problem to the next, as long as these constraints remain valid. This can be implemented quite easily when the branch-and-cut algorithm already maintains a cut-pool. Typically, the latter contains inequalities that have been removed to reduce the size of the model. Since previously removed cuts can be violated again later on, they are kept in a pool which is explored at the beginning of the separation phase. We thus suggest to initialize the cut-pool of $P_i(\epsilon_j^s)$ with the active cuts at the optimum of $P_i(\epsilon_j^{s-1})$ (another way to manage a cut-pool is proposed in [25] for the Bi-objective Traveling Purchaser problem). Knowledge of the polytope can also be exploited to generate valid inequalities. In the next section, we describe an inequality for the TSPP based on the optimal solution of the previous problem.

Improving the initial feasible solution . Similarities between consecutive solutions can be exploited by the heuristic that generates the initial solution. A better initial solution increases the upper bound on the optimal solution value and thus allows to prune branches early and reduce the number of explored nodes.

3 The Traveling Salesman Problem with Profits

To empirically validate our ϵ-constraint method for BOCO problems, we applied it to the TSPP. This section describes the problem and explains how our ϵ-constraint method can solve it.

3.1 Problem description

Among the multiple variants of the TSP [14], the TSPP belongs to the *selective TSP* class where a feasible solution is not required to visit all vertices. We use the classification in [11], where variants of the selective TSP in which values are associated with vertices are considered to be TSPP. The latter is a BOCO problem where two strongly conflicting objectives must be optimized. Namely, one must find a Hamiltonian cycle over a subset of vertices such that the collected prize is maximized while the travel cost is minimized. The prize collection maximization implies that the traveler should visit a large number of vertices, while the cost minimization has the opposite effect.

The scope of our discussion will be restricted to the undirected TSPP which can be mathematically formulated as follows. Let $G = (V, E)$ be an undirected complete graph, with edge set E and vertex set V, among which vertex 1 stands for the depot. The nonnegative integer prizes are denoted p_v for each $v \in V$ ($p_1 = 0$). For every $e \in E$, the nonnegative integer travel cost c_e satisfies the triangle inequality. We define $E(S) = \{(u, v) \in E : u \in S,\ v \in S\}$ and $\delta(S) = \{(u, v) \in E : u \in S,\ v \notin S\}$ for $S \subset V$, and $V(T) = \{v \in V : T \cap \delta(\{v\}) \neq \emptyset\}$ for $T \subseteq E$. We also define, $V' = V \setminus \{1\}$ and for each $v \in V$, we write $\delta(v)$ instead of $\delta(\{v\})$. The decision variables are :

$$x_e = \begin{cases} 1 & \text{if edge } e \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

$$y_v = \begin{cases} 1 & \text{if vertex } v \text{ is visited} \\ 0 & \text{otherwise} \end{cases}$$

Finally, we define $y(S) = \sum_{v \in S} y_v$ and $p(S) = \sum_{v \in S} p_v$ for $S \subseteq V$, and $x(T) = \sum_{e \in T} x_e$ and $c(T) = \sum_{e \in T} c_e$ for $T \subseteq E$. The TSPP can then be formulated as the following 0-1 integer linear program (LP):

$$\max \sum_{v \in V} p_v y_v \tag{10}$$

$$\min \sum_{e \in E} c_e x_e \tag{11}$$

subject to:

$$x(\delta(v)) = 2y_v \qquad (\forall\ v \in V) \tag{12}$$

$$x(\delta(S)) \geq 2y_v \qquad (\forall\ S \subset V :\ 1 \in S,\ v \in V \setminus S) \tag{13}$$

$$y_1 = 1 \tag{14}$$

$$x_e \in \{0, 1\} \qquad (\forall\ e \in E) \tag{15}$$

$$y_v \in \{0, 1\} \qquad (\forall\ v \in V') \tag{16}$$

The degree constraints (12) insure that a feasible solution goes exactly once through each visited vertex. The subtour elimination constraints (13) require that each visited vertex $v \in V'$ of a feasible solution be

reachable from the depot by two edge-disjoint paths. Constraint (14) forces the depot to be visited and constraints (15) and (16) impose that all variables be 0-1. Note that the model forces all feasible solutions to visit at least 3 vertices. Since solutions with less than 3 vertices can be easily found by explicit enumeration, we assume, without loss of generality, that optimal solutions contain at least 3 vertices.

In spite of its bi-objective nature, the literature focuses on mono-criterion variants of the TSPP. Three variants have been extensively studied up to now: the Profitable Tour Problem (PTP), the Orienteering Problem (OP) and the Prize Collecting TSP (PCTSP). The PTP, introduced in [8], minimizes the difference between the collected prizes and the travel cost; it is also known as the Simple Cycle Problem [13]. In the OP, one must find a tour that maximizes the total collected prize while maintaining the traveling cost under a fixed value. It has been introduced in a study on orienteering competitions [28] and it is also known as the Selective TSP [20] and as the Maximum Collection Problem [17]. Finally, the PCTSP was introduced as a model for scheduling the daily operations of a steel rolling mill [3]. Given an undirected graph with edge costs and node prizes, the aim of the PCTSP is to find a simple cycle minimizing the total edge cost while collecting a minimum total prize. The PCTSP is also known as the Quota TSP [1].

The bi-criteria nature of the TSPP has been considered in [18] where the efficient set is approximated for small problems with less than 25 vertices. To the best of our knowledge, the bi-objective TSPP has never been solved exactly. However, there is some literature on exact algorithms for mono-criterion variants of the TSPP which are mostly adaptations of branch-and-bound procedures developed for the TSP (see [11] for a complete survey).

3.2 Finding the exact Pareto front of the TSPP

The TSPP, as defined by equations (10) to (16), is a BOCO problem where both objectives take integer values. Theorem 3 shows how to apply the ϵ-constraint method to find the exact Pareto front of such problems. One must first decide which of the cost minimization or the collected prize maximization should be kept in the objective. In other words, one should decide if the ϵ-constraint problems will be OPs (collected prize maximization) or PCTSPs (cost minimization). The PCTSP is clearly the best choice because the greatest common divisor among the prize values is often greater than one. That is, Δ can often take a value greater than one (see Proposition 1) which strengthens the minimum prize constraint by forcing its upper bound (ϵ) to be as close as possible to the collected prize of the optimal PCTSP solution. Note that it is not possible to set $\Delta > 1$ in the OP, since one can hardly find the greatest common divisor among all possible Hamiltonian cycle costs.

The mathematical formulation of the PCTSP is similar to the one for the TSPP (equations 10 to 16), except for the first objective (10) which is replaced by the minimum collected prize constraint:

$$\sum_{v \in V} y_v p_v \geq \bar{p} \tag{17}$$

where p_v is the prize associated with vertex v and $\bar{p}$ is a constant corresponding to the minimum prize to be

collected. Equation (17) can also be formulated as

$$\sum_{v \in V} p_v(1 - y_v) \leq U \tag{18}$$

where $U = p(V) - \bar{p}$. This corresponds to the ϵ-constraint of $P_i(\epsilon_j)$, where ϵ_j is represented by U. The sequence of PCTSPs starts with the worst $\bar{p}$ value ($\bar{p} = 0$). The latter is then progressively increased until the largest possible value ($\bar{p} = p(V)$) is reached. Note that when $\bar{p}$ is null, the optimal PCTSP solution stays at the depot. This zero cost solution corresponds to the point (c^I, p^N). On the other hand, the point (c^N, p^I) is associated with the PCTSP with $\bar{p} = p(V)$, which is a classical TSP.

The PCTSPs are efficiently solved through the branch-and-cut procedure described in [5]. The latter has been modified to generate a solution with the greatest collected prize among the optimal solutions of $P_i(\epsilon_j)$, as explained under *Handling dominated points* in Section 1. Thereby, the number of PCTSPs to be solved is limited to the number of points on the Pareto front. Our experiments show that this actually reduces the number of subproblems by 10% on average. However, because those subproblems are harder to solve, we observe an increase of 159% in computation time, on average. We thus decided to use the original version of the branch-and-cut algorithm, since it is more efficient overall.

From all the valid PCTSP inequalities used in this algorithm (see Appendix A), only the cost-cover and conditional inequalities are not guaranteed to be valid for any ϵ_j. Both cuts are based on a set of vertices or on a cycle that leads to solutions with a cost greater than the best known feasible solution value. Since the sequence of $P_i(\epsilon_j)$ problems is such that the solution values increase, the upper bound for a subproblem is no longer valid for the next subproblems. That is, we do not separate the cost-cover and conditional inequalities. The lifted-cover inequalities are still valid in subsequent subproblems, although they lose their strength because the cover might not remain minimal. Cycle-cover inequalities also remain valid because the minimum collected prize increases from one subproblem to the next. Simple comb, 2-matching and logical inequalities are clearly valid for the whole sequence of PCTSPs.

The computation time of the branch-and-cut algorithm that solves the PCTSPs can be substantially reduced by using the heuristic improvements presented in Section 2. In the following, we show how those heuristics have been implemented.

Exploiting knowledge of the polytope. In addition to reusing previously separated inequalities, the knowledge of the polytope can be improved by adding the following inequality. Since the minimum collected prize increases from one subproblem to the next, at least one unvisited vertex in $P_i(\epsilon_j^s)$ must be visited in $P_i(\epsilon_j^{s+1})$. Hence, we have the following valid *visit inequality*:

$$y(V \setminus V_s^*) \geq 1 \tag{19}$$

where V_s^* is the set of visited vertices in an optimal solution of $P_i(\epsilon_j^s)$. Such an inequality is added to the LP of each new subproblem.

Improving the initial feasible solution . A good feasible solution can be obtained through a modification to the optimal solution of the previous ϵ-constraint problem. Adding any vertex to the optimal solution of $P_i(\epsilon_j^{s-1})$ actually produces a feasible solution to $P_i(\epsilon_j^s)$. Let $\vec{x}_{s-1}^*$ be an optimal solution to $P_i(\epsilon_j^{s-1})$. Then, the following heuristic procedure generates a feasible solution $\tilde{x}_s$ to $P_i(\epsilon_j^s)$:

1. Compute a feasible solution $\tilde{x}_s$ with the heuristic algorithm used in [5] to find an initial feasible solution.
2. For every vertex $v \in V \setminus V_{s-1}^*$:
   a) Construct solution $\hat{x}_s$ by inserting v in $\vec{x}_{s-1}^*$ at a location that minimizes the detour.
   b) If $c(\tilde{x}_s) > c(\hat{x}_s)$, set $\tilde{x}_s = \hat{x}_s$.

4 Computational results

We transformed VRP and TSP instances of the TSPLIB [24] into TSPP instances using the rules provided in [5] and [12]. We considered instances for which the node coordinates were available. For VRP instances, the demands are interpreted as the node prizes. For TSP instances, the prizes p_v $(v \in V')$ are generated in three different ways:

- Generation 1: $p_v = 1$;
- Generation 2: $p_v = 1 + (7141v + 73) \mod 100$;
- Generation 3: $p_v = 1 + \lfloor 99\frac{c_{1,v}}{\theta} \rfloor$, where $\theta = \max_{w \in V'} c_{1,w}$.

Instances of generation 1 are in general easy problems since all prizes are the same. Generation 2 produces instances with pseudo-random prizes while generation 3 produces hard problems where larger prizes are associated with vertices that are further from the depot. The algorithm was implemented in C++ and was run on a AMD Opteron 2.4 Ghz processor. The LPs were solved using CPLEX 9.3.

This section starts with the performance analysis of our improvement heuristics before showing results for all solved instances. Finally, we analyse the quality of the approximate Pareto fronts obtained, as explained in Section 1.

4.1 Performance of the improvement heuristics

Table 1 shows data on the performance of the improvement heuristics on a representative sample of instances. The computation times, in seconds, of the standard algorithm without any improvement heuristic are given in column *STD*. The three next columns give the relative improvement (in percent) of the computation time due to each of the three improvement heuristics:

- *ACH*: keep active cuts for subsequent subproblems,
- *VSI*: visit inequalities,

- *ISH*: initial solution heuristic.

The columns *ALL* and *AIH* give the computation time and the relative improvement, respectively, when all three improvement heuristics are enabled. For each instance, the best improvement is identified in bold face.

The results show that keeping the active inequalities from one PCTSP to the next significantly reduce the computation time (33% in average). We have no strong evidence that the visit inequalities reduce the computation time, although a small improvement is observed on average. On 5 instances, however, the introduction of visit inequalities slightly increases the computation time. Since the improvement is sometimes around 10% while the computation times never increase by more than 3%, we decided to keep them for the exhaustive tests reported later. Using the solution of the previous PCTSP to find a feasible initial solution for the next subproblem improves the computation time by 1.4% to 24.9% (9.18% in average). Combining the three heuristics almost always give the best results. Two exceptions are reported in Table 1 where one should have used only the *ACH* heuristic.

Instance	Type	STD	ACH	VSI	ISH	ALL	AIH
eil33	vrp	85.41	24.39	3.47	11.57	54.89	**35.73**
eilA76	vrp	4528.03	20.88	12.04	23.14	2605.27	**42.46**
att48	tspp1	19.08	46.54	-1.00	2.31	9.82	**48.53**
eil76	tspp1	105.43	52.74	-0.16	24.92	37.19	**64.73**
rd100	tspp1	829.61	53.61	-0.85	7.32	365.53	**55.94**
pr144	tspp1	40002.63	50.63	8.06	15.47	17481.92	**56.3**
ch150	tspp1	10362.8	46.65	1.51	16.66	3706.72	**64.23**
ulysses22	tspp2	21.65	30.02	11.69	3.19	13.6	**37.18**
att48	tspp2	1215.76	15.14	2.08	2.00	866.16	**28.76**
berlin52	tspp2	1386.49	**20.21**	2.47	6.54	1321.63	4.68
eil76	tspp2	6508.44	18.26	2.47	8.3	5158.31	**20.74**
rd100	tspp2	79563.53	41.23	-0.81	6.88	43100.31	**45.83**
ulysses22	tspp3	21.27	28.40	8.04	1.41	14.04	**33.99**
att48	tspp3	1788.56	14.66	-3.01	1.67	1334.41	**25.39**
berlin52	tspp3	2276.94	24.77	0.79	1.88	1292.27	**43.25**
st70	tspp3	17528.77	38.02	0.17	5.75	9258.1	**47.18**
eil76	tspp3	7986.51	**40.52**	4.14	17.1	4971.95	37.75
Averages			33.33	3.01	9.18		**40.75**

Table 1: Performance of the improvement heuristics

4.2 Results for exact Pareto fronts

Tables 2, 3, 4, and 5 show results for VRP instances and TSP generation 1, 2 and 3 instances, respectively. The three improvement heuristics were enabled for each instance reported. The columns correspond to:

- TIME: the total computation time, in seconds;
- $|\mathcal{F}|$: the size of the Pareto front;
- N: the number of ϵ-constraint problems solved[2];

[2]The ϵ-constraint problems are PCTSPs that visit at least 3 vertices. The trivial solutions containing 1 or 2 vertices are found by enumeration and are therefore not included in N.

- $\bar{t}$: the average computation time of the ϵ-constraint problems;
- σ_t: the standard deviation of the ϵ-constraint problems' computation time;
- L-5%: the percentage of the computation time spent on the 5% harder problems;
- S-50%: the percentage of the computation time spent on the 50% easier problems.

The letters *t.l.* (time limit) indicates that the instance was still unsolved after a time limit of 72 hours (259,200 seconds). For those instances, the column $|\mathcal{F}|$ gives the ratio $\alpha = \frac{\bar{p}}{p(V)}$, which is an indication of the portion of the Pareto front that has been found before the time limit was reached.

Our algorithm was able to solve instances of 150 vertices for easy instances (TSP generation 1) and up to about 100 vertices for harder instances. Among those that remained unsolved after 72 hours, 39% were almost solved ($\alpha > 0.8$) while there was still a lot to do for 28% of them ($\alpha < 0.2$). Observe that the latter are all very hard instances (TSP generation 3).

Two factors characterize hard TSPP instances: the size of the Pareto front and the difficulty of the subproblems (PCTSPs). An empirical evidence of the first factor is a correlation coefficient of 0.74 between the size of the Pareto front and the total computation time. We observed that all instances solved in more than 24 hours are from generations 2 and 3 (except for one), which is not a surprise since both generations are designed to produce instances with a lot of efficient solutions. On average, the ratio of $|\mathcal{F}|$ over the number of vertices is 8.08 and 7.84 for generations 2 and 3, respectively, while it is 1.0 for generation 1 and 3.78 for VRP instances.

The impact of the subproblems' toughness on the computation time is partially shown by a correlation coefficient of 0.51 between the average subproblem computation time and the total computation time. Although this correlation is significant, it does not tell the whole story. One should observe that the subproblems are not equally hard. In fact, the computation times are mostly due to a few PCTSPs. This phenomenon is observed on hard instances with a relatively small Pareto front (that is, the latter cannot explain the instance's toughness), such as *pr76, kroA150, kroB150* and *pr136* generation 1 instances. The σ_t statistic of those instances is very high and 40% to 83% of the computation time is spent on 5% of the subproblems. Moreover, less than 2% of the computation is spent on 50% of the subproblems (0.02% for *pr76* instance).

Instance	TIME	$\lvert\mathcal{F}\rvert$	N	$\bar{t}$	σ_t	L-5%	S-50%
eil22	7.96	67	71	0.11	0.05	9.67	32.04
eil23	8.37	75	77	0.11	0.07	9.8	21.74
eil30	27.3	125	141	0.19	0.18	18.79	19.08
eil33	54.89	159	228	0.24	0.21	16.82	20.13
att48	11.47	48	47	0.24	0.2	15.34	22.58
eil51	539.6	223	254	2.12	2.28	21.65	16.36
eilA76	2605.27	355	458	5.69	7.15	24.62	14.6
eilA101	5046.78	498	701	7.2	9.7	27.13	10.39
gil262	t.l.	29.48	-	-	-	-	-

Table 2: Statistics on the VRP instances

Instance	TIME	$\|\mathcal{F}\|$	N	$\bar{t}$	σ_t	L-5%	S-50%
burma14	0.14	14	12	0.01	0.01	21.43	35.71
ulysses16	0.2	16	14	0.01	0.01	15.0	25.0
ulysses22	0.51	22	21	0.02	0.01	11.76	25.49
att48	9.82	48	47	0.21	0.15	14.56	24.75
eil51	10.53	51	50	0.21	0.29	23.84	13.49
berlin52	10.18	52	51	0.2	0.17	13.16	18.86
st70	61.57	70	69	0.89	1.71	36.09	6.11
eil76	37.19	76	75	0.5	0.59	21.65	18.93
pr76	175334.47	76	75	2337.79	10774.27	83.3	0.02
rat99	233.78	99	98	2.39	3.03	23.57	16.26
kroA100	341.15	100	99	3.45	4.81	23.22	9.18
kroB100	1075.63	100	99	10.86	33.79	54.69	2.55
kroC100	303.73	100	99	3.07	3.58	19.94	14.0
kroD100	178.9	100	99	1.81	2.53	23.41	11.1
kroE100	837.43	100	99	8.46	25.39	52.05	3.42
rd100	365.53	100	99	3.69	10.8	59.36	5.01
eil101	90.37	101	100	0.9	0.84	19.58	20.52
lin105	5558.73	105	104	53.45	152.07	61.17	1.66
pr107	74.12	107	106	0.7	0.76	20.67	16.99
pr124	2990.82	124	123	24.32	44.57	34.47	3.32
bier127	1073.62	127	126	8.52	18.13	43.96	6.68
ch130	719.24	130	129	5.58	12.87	42.61	6.25
pr136	64590.76	136	135	478.45	1627.46	68.69	0.13
gr137	3354.58	137	136	24.67	64.72	48.83	3.95
pr144	17481.92	144	143	122.25	675.57	80.94	1.56
ch150	3706.72	150	149	24.88	51.03	37.53	3.14
kroA150	81024.71	150	149	543.79	1066.54	39.86	1.52
kroB150	68089.94	150	149	456.98	1309.99	54.31	0.39
pr152	t.l.	*93.38*	-	-	-	-	-
u159	t.l.	*39.87*	-	-	-	-	-

Table 3: Statistics on the TSP generation 1 instances

Instance	TIME	$\|\mathcal{F}\|$	N	$\bar{t}$	σ_t	L-5%	S-50%
burma14	1.73	59	60	0.03	0.02	15.03	27.17
ulysses16	4.31	102	101	0.04	0.02	10.21	34.8
ulysses22	13.6	130	130	0.1	0.05	9.78	32.65
att48	866.16	435	438	1.98	1.21	13.1	26.85
eil51	627.13	225	269	2.33	2.26	19.24	16.81
berlin52	1321.63	406	411	3.22	2.75	15.97	16.84
st70	13892.93	503	643	21.61	38.21	34.48	4.53
eil76	5158.31	386	538	9.59	10.81	22.45	16.28
pr76	t.l.	*96.29*	-	-	-	-	-
rat99	31524.89	662	779	40.47	76.59	38.31	11.11
kroA100	t.l.	87.88	-	-	-	-	-
kroB100	186395.45	1332	1363	136.75	337.95	50.42	7.33
kroC100	120664.66	1311	1333	90.52	129.24	29.95	18.99
kroD100	53819.04	1128	1129	47.67	33.95	15.43	25.2
kroE100	82149.58	1068	1086	75.64	119.62	32.46	9.26
rd100	43100.31	920	962	44.8	30.01	14.31	26.92
eil101	34953.71	515	838	41.71	45.84	21.87	13.5
lin105	203727.18	1043	1329	153.29	291.15	36.25	10.71
pr107	t.l.	*49.36*	-	-	-	-	-
pr124	t.l.	*29.24*	-	-	-	-	-
bier127	t.l.	*87.65*	-	-	-	-	-
ch130	t.l.	*93.87*	-	-	-	-	-

Table 4: Statistics on the TSP generation 2 instances

Instance	TIME	$\|\mathcal{F}\|$	N	$\bar{t}$	σ_t	L-5%	S-50%
burma14	1.99	70	68	0.03	0.01	9.05	31.16
ulysses16	3.81	92	88	0.04	0.02	9.45	32.28
ulysses22	14.04	128	126	0.11	0.05	10.19	31.98
att48	1334.41	438	440	3.03	1.83	13.01	26.95
eil51	1196.46	267	299	4.0	6.16	29.71	10.51
berlin52	1292.27	439	446	2.9	2.12	15.61	24.26
st70	9258.1	452	546	16.96	17.35	19.27	12.52
eil76	4971.95	383	468	10.62	8.87	16.68	20.9
pr76	t.l.	*82.52*	-	-	-	-	-
rat99	t.l.	*5.70*	-	-	-	-	-
kroA100	137168.33	815	820	167.28	209.37	25.77	13.12
kroB100	t.l.	*16.75*	-	-	-	-	-
kroC100	128195.9	1223	1228	104.39	106.56	20.87	17.14
kroD100	71826.93	1063	1068	67.25	73.97	20.6	24.61
kroE100	t.l.	*10.38*	-	-	-	-	-
rd100	180959.84	1513	1551	116.67	428.34	65.3	6.9
eil101	38227.19	499	697	54.85	92.39	33.78	7.62
lin105	t.l.	*9.77*	-	-	-	-	-
pr107	t.l.	*2.16*	-	-	-	-	-
pr124	t.l.	*3.66*	-	-	-	-	-
bier127	t.l.	*16.09*	-	-	-	-	-
ch130	t.l.	*88.35*	-	-	-	-	-

Table 5: Statistics on the TSP generation 3 instances

4.3 Results for approximate Pareto fronts

As explained in Section 1, our algorithm can produce an approximation of the Pareto front. One simply has to introduce a tolerance (ρ) on the minimal gap between the upper and lower bound for a solution to be accepted by the branch-and-cut procedure. Tables 6 and 7 show results for ρ of 0.01 and 0.10, respectively, on a sample of hard instances for which the exact Pareto front is known.

There is no consensus on the quality metrics that should be used for multi-criteria approximation algorithms. We decided to use two categories of metrics reported in [15]. The first category is made of distance based metrics while the second is made of ratios on the size of the exact and approximated Pareto fronts. More precisely, the columns of Tables 6 and 7 correspond to:

- t_e : computation time for the exact Pareto front ($\mathcal{F}$);
- t_a : computation time for the approximate Pareto front ($\bar{\mathcal{F}}$);
- $d_p = \frac{1}{|\mathcal{F}|} \sum_{z \in \mathcal{F}} \min_{z' \in \bar{\mathcal{F}}} \frac{|z_p - z'_p|}{z_p}$, where z_p stands for the collected prize associated with point z;
- $d_c = \frac{1}{|\mathcal{F}|} \sum_{z \in \mathcal{F}} \min_{z' \in \bar{\mathcal{F}}} \frac{|z_c - z'_c|}{z_c}$, where z_c stands for the travel cost associated with point z;
- $d_z = \frac{1}{|\mathcal{F}|} \sum_{z \in \mathcal{F}} \min_{z' \in \bar{\mathcal{F}}} \max\left(\frac{|z_p - z'_p|}{z_p}, \frac{|z_c - z'_c|}{z_c}\right)$[3];
- $d_z^{max} = \max_{z \in \mathcal{F}} \min_{z' \in \bar{\mathcal{F}}} \max\left(\frac{|z_p - z'_p|}{z_p}, \frac{|z_c - z'_c|}{z_c}\right)$;
- $Q_1 = \frac{\mathcal{F} \cap \bar{\mathcal{F}}}{|\mathcal{F}|}$;
- $Q_2 = \frac{\mathcal{F} \cap \bar{\mathcal{F}}}{|\bar{\mathcal{F}}|}$.

For $\rho = 0.01$, the algorithm is 5.7 times faster than the original exact version. It produces a very good approximation of the Pareto front for every instance of the sample. Even though only an average of 38% of the non dominated points are found, each non dominated point is on average at a distance of 0.002 of a point on the approximated front, according to the relative Chebyshev distance metric. When the tolerance is increased to 0.10, the algorithm runs 83 times faster than the original version and still finds a good approximation of the Pareto front. The average relative Chebyshev distance between each point of $\mathcal{F}$ and the nearest point of $\bar{\mathcal{F}}$ is 0.018.

One should observe that for both values of ρ, Q_1 is always lower than Q_2. For example, an average of 38.0% and 62.3 % of the points on $\mathcal{F}$ are also on $\bar{\mathcal{F}}$ when $\rho = 0.01$ and $\rho = 0.10$, respectively. This suggests that the approximated non dominated set is smaller than the exact one. Actually, the size of the approximated front corresponds to 50.5% ($\rho = 0.01$) and 22.3% ($\rho = 0.10$) of $|\mathcal{F}|$, on average. Those averages exclude TSP generation 1 instances for which both the exact and approximate fronts have about the same size. Finally, although there is no theoretical guarantee on the performance of the approximation algorithm, the results show relatively small variations in d_z. This suggests that the strategy is quite robust.

[3]This corresponds to an average of the best relative Chebyshev distances. The latter defines the distance between the points (x_1, y_1) and (x_2, y_2) as: $\max(|x_2 - x_1|, |y_2 - y_1|)$.

Instance	Type	t_e	t_a	t_a/t_e	d_p	d_c	d_z	d_z^{max}	Q_1	Q_2
eilA76	vrp	2605.27	734.63	0.282	0.001	0.002	0.002	0.026	0.603	0.903
eilA101	vrp	5046.78	1847.79	0.366	0.001	0.001	0.001	0.010	0.538	0.736
pr76	tspp1	175334.47	28747.08	0.164	0.000	0.002	0.002	0.010	0.461	0.461
pr136	tspp1	64590.76	4742.81	0.073	0.000	0.001	0.002	0.009	0.500	0.504
pr144	tspp1	17481.92	5819.35	0.333	0.000	0.001	0.002	0.019	0.465	0.479
kroA150	tspp1	81024.71	2516.27	0.031	0.000	0.002	0.003	0.010	0.253	0.253
kroB150	tspp1	68089.94	22744.47	0.334	0.000	0.002	0.003	0.010	0.407	0.407
st70	tspp2	13892.93	1061.9	0.076	0.001	0.001	0.002	0.018	0.475	0.685
kroB100	tspp2	186395.45	8089.15	0.043	0.001	0.001	0.002	0.029	0.261	0.645
kroC100	tspp2	120664.66	5988.84	0.050	0.001	0.001	0.002	0.041	0.250	0.659
rd100	tspp2	43100.31	3712.49	0.086	0.001	0.001	0.002	0.032	0.262	0.623
lin105	tspp2	203727.18	31861.07	0.156	0.001	0.001	0.002	0.081	0.320	0.742
st70	tspp3	9258.1	2192.75	0.237	0.002	0.001	0.003	0.032	0.473	0.751
kroA100	tspp3	137168.33	17569.18	0.128	0.002	0.001	0.003	0.050	0.329	0.786
kroD100	tspp3	71826.93	12683.36	0.177	0.002	0.001	0.003	0.039	0.249	0.639
rd100	tspp3	180959.84	33950.47	0.188	0.002	0.001	0.003	0.143	0.171	0.617
eil101	tspp3	38227.19	10434.01	0.273	0.001	0.002	0.002	0.017	0.439	0.709
Averages				**0.176**	**0.001**	**0.001**	**0.002**	**0.034**	**0.380**	**0.623**

Table 6: Results for approximate Pareto fronts with $\rho = 0.01$

Instance	Type	t_e	t_a	t_a/t_e	d_p	d_c	d_z	d_z^{max}	Q_1	Q_2
eilA76	vrp	2605.27	22.83	0.009	0.007	0.008	0.019	0.211	0.085	0.345
eilA101	vrp	5046.78	83.48	0.017	0.005	0.004	0.014	0.141	0.084	0.228
pr76	tspp1	175334.47	17.67	0.000	0.002	0.008	0.028	0.083	0.118	0.122
pr136	tspp1	64590.76	354.62	0.005	0.001	0.005	0.017	0.070	0.206	0.215
pr144	tspp1	17481.92	599.07	0.034	0.002	0.007	0.018	0.050	0.139	0.157
kroA150	tspp1	81024.71	164.27	0.002	0.002	0.006	0.020	0.083	0.073	0.078
kroB150	tspp1	68089.94	367.13	0.005	0.000	0.006	0.018	0.083	0.127	0.129
st70	tspp2	13892.93	39.33	0.003	0.007	0.006	0.018	0.113	0.074	0.272
kroB100	tspp2	186395.45	163.33	0.001	0.004	0.004	0.014	0.178	0.035	0.208
kroC100	tspp2	120664.66	224.53	0.002	0.006	0.005	0.015	0.155	0.039	0.291
rd100	tspp2	43100.31	1623.16	0.009	0.006	0.003	0.014	0.243	0.052	0.226
lin105	tspp2	203727.18	528.95	0.003	0.005	0.005	0.025	0.068	0.023	0.122
st70	tspp3	9258.1	460.18	0.050	0.007	0.004	0.022	0.143	0.084	0.297
kroA100	tspp3	137168.33	1211.63	0.009	0.008	0.005	0.015	0.356	0.058	0.333
kroD100	tspp3	71826.93	1894.98	0.026	0.006	0.004	0.016	0.125	0.027	0.149
rd100	tspp3	180959.84	1623.16	0.009	0.006	0.003	0.013	0.143	0.032	0.214
eil101	tspp3	38227.19	1805.84	0.047	0.007	0.006	0.019	0.113	0.066	0.234
Averages				**0.012**	**0.005**	**0.006**	**0.018**	**0.138**	**0.078**	**0.213**

Table 7: Results for approximate Pareto fronts with $\rho = 0.10$

Conclusion

We have shown that the ϵ-constraint method can be used efficiently to find the exact Pareto front of BOCO problems with integer objective values. We also provide improvement heuristics devised to speed up the resolution of the ϵ-constraint problems when the latter are solved through branch-and-cut. Our ϵ-constraint method and the improvement heuristics have been tested successfully on the TSPP. The results have shown

the relevance of the improvement heuristics and provided the first exact solutions for TSPP instances. Because the TSPP is a very hard problem, the instances that have been solved are quite small. Obviously, exact algorithms cannot run very fast on BOCO problems, but we believe that our solutions will be useful benchmarks to evaluate the quality of future approximation algorithms for the TSPP. Besides, we have shown that good approximations of the Pareto front might be found relatively quickly through a simple modification of our exact algorithm.

Acknowledgments

Financial support for this work was provided by the Canadian Natural Sciences and Engineering Research Council (NSERC) and by the Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT). This support is gratefully acknowledged.

References

[1] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. New approximation guarantees for minimum-weight k-trees and prize-collecting salesman. *SIAM Journal on Computing*, 28:254–262, 1998.

[2] E. Balas. Facets of the knapsack polytope. *Mathematical Programming*, 8:146–164, 1975.

[3] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[4] E. Balas. The prize collecting traveling salesman problem: II. Polyhedral results. *Networks*, 25:199–216, 1995.

[5] J.-F. Bérubé, M. Gendreau, and J.-Y. Potvin. A branch-and-cut algorithm for the undirected prize collecting traveling salesman problem. Technical Report CRT-2006-30, Centre for Research on Transportation, Université de Montréal, 2006.

[6] V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making: Theory and Methodology*. North-Holland, 1983.

[7] J. C. N. Clímaco and E. Q. V. Martins. A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11:399–404, 1982.

[8] M. Dell'Amico, F. Maffioli, and P. Värbrand. On prize-collecting tours and the asymmetric travelling salesman problem. *International Transactions in Operational Research*, 2:297–308, 1995.

[9] M. Ehrgott and X. Gandibleux. Multiobjective combinatorial optimization - theory, methodology, and applications. In M. Ehrgott and X. Gandibleux, editors, *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, pages 369–444. Kluwer Academic Publishers, 2002.

[10] M. Ehrgott and M. M. Wiecek. Multiobjective programming. In J. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple Criteria Decision Analysis: State of the Art Surveys*, pages 667–722. Springer's International Series, 2005.

[11] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39:188–205, 2005.

[12] M. Fischetti, J. J. Salazar-González, and P. Toth. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, 10:133–148, 1998.

[13] M. Fischetti, J. J. Salazar-González, and P. Toth. The generalized traveling salesman and orienteering problems. In G. Gutin and A. P. Punnen, editors, *The Traveling Salesman Problem and its Variations*, pages 609–662. Kluwer Academic Publishers, 2002.

[14] G. Gutin and A. P. Punnen, editors. *The Traveling Salesman Problem and its Variations*. Kluwer Academic Publishers, 2002.

[15] A. Jaszkiewicz. Evaluation of multiple objective metaheuristics. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimization*, pages 65–89. Springer-Verlag, 2004.

[16] N. Jozefowiez, F. Semet, and E.-G. Talbi. The bi-objective covering tour problem. *Computers & Operations Research*, In press.

[17] S. Kataoka and S. Morito. An algorithm for single constraint maximum collection problem. *Journal of the Operations Research Society of Japan*, 31:515–530, 1988.

[18] C. P. Keller and M. F. Goodchild. The multiobjective vending problem: a generalization of the travelling salesman problem. *Environment and Planning B: Planning and Design*, 15:447–460, 1988.

[19] P. Kouvelis and S. Sayin. Algorithm robust for the bicriteria discrete optimization problem: Heuristic variations and computational evidence. *Annals of Operations Research*, 147:71–85, 2006.

[20] G. Laporte and S. Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 26:193–207, 1990.

[21] M. Laumanns, L. Thiele, and E. Zitzler. An efficient, adaptative parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169:932–942, 2006.

[22] K. M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer academic, 1999.

[23] T. K. Ralphs, M. J. Saltzman, and M. M. Wiecek. An improved algorithm for solving biobjective integer programs. *Annals of Operations Research*, 147:43–70, 2006.

[24] G. Reinelt. TSPLIB, A traveling salesman problem library. *ORSA Journal on Computing*, 3:376–384, 1991.

[25] J. Riera-Ledesma and J. J. Salazar-González. The biobjective travelling purchaser problem. *European Journal of Operational Research*, 160:599–613, 2005.

[26] S. Sayin and P. Kouvelis. The multiobjective discrete optimization problem: a weighted min-max two-stage optimization approach and a bicriteria algorithm. *Management Science*, 51:1572–1581, 2005.

[27] R. E. Steuer, L. R. Gardiner, and J. Gray. A bibliographic survey of the activities and international nature of multiple criteria decision making. *Journal of Multi-Criteria Decision Analysis*, 5:195–217, 1996.

[28] T. Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35:797–809, 1984.

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

A Appendix - Valid inequalities for the PCTSP

This appendix summarizes the valid inequalities used by the branch-and-cut algorithm presented in [5] for the model defined by equations (11) to (17). They are obtained either from the associated knapsack polytope, from a combination of the SECs and the minimum prize constraint, or from the associated traveling salesman polytope.

A.1 Inequalities from the associated knapsack polytope

Two types of inequalities based on knapsack constraints are considered. They are referred as the *lifted-cover* and *cost-cover* inequalities.

A.1.1 Lifted-cover inequalities

Let S be a minimal cover for (18), i.e., S is a minimal subset of V such that $p(S) > U$. The cover inequality:

$$\sum_{v \in (S \cup S')} (1 - y_v) \leq |S| - 1 \tag{20}$$

where $S' = \{v \in V \setminus S : p_v \geq \max_{w \in S} p_w\}$, is valid for the knapsack problem [2]. The coefficients of the y variables can be lifted to obtain *lifted-cover* inequalities that reinforce equation (20). Let $S' = \{v \in$

$V \setminus S : p_v \geq \max_{w \in S} p_w\}$, and S_h the set of the first h elements of S ($\forall i \in S$, $p_i \geq p_{i+1}$ is assumed), $h = 1, ..., |S|$. Let V be partitioned into $V_0, V_1, .., V_q$, $q = |S| - 1$, where:

$$\begin{aligned} V_h = \left\{v \in (S \cup S') : p(S_h) \leq p_v < p(S_{h+1})\right\}, \quad h = 2, ..., q \\ V_1 = (S \cup S') \setminus \cup_{h=2}^{q} V_h, \\ V_0 = V \setminus (S \cup S') \end{aligned} \tag{21}$$

and define:

$$\pi_v = h, \quad \forall v \in V_h, \quad h = 0, ..., q. \tag{22}$$

Then, the lifted-cover inequality is written:

$$\sum_{v \in S} (1 - y_v) + \sum_{v \in V \setminus S} \pi_v (1 - y_v) \leq |S| - 1 \tag{23}$$

It has been shown that (23) is valid for all $y \in KP$, where KP is the convex hull of $\{y \in \{0, 1\} : y$ satisfies (18) $\}$ [2]. Since the PCTS polytope is included in KP [3], the lifted-cover inequalities are also valid for the PCTSP.

A.1.2 Cost-cover inequalities

Let c_U be the upper bound on an optimal solution. Then $\sum_{e \in E} c_e x_e \leq c_U$ defines a knapsack constraint in terms of costs that can be used to derive valid inequalities. Let $S \subseteq V$, $1 \in S$ and σ_S a lower bound on the optimal TSP value on S. Then, if $\sigma_S > c_U$ and if the costs satisfy the triangle inequality, the following *cost-cover* inequalities are valid for the PCTSP. We consider only special cases that are easy to separate, namely, when $|S| = 3$:

$$y_u + y_v \leq 1 \quad \forall u, v \in V' \quad \text{such that} \quad c_{(1,u)} + c_{(u,v)} + c_{(v,1)} > c_U \tag{24}$$

and when $|S| = 2$:

$$y_v = 0 \quad \forall v \in V' \quad \text{such that} \quad 2c_{(1,v)} > c_U \tag{25}$$

A.2 Inequalities from the SEC and knapsack constraint

Cycle-cover and *conditional* inequalities both use a knapsack constraint to strengthen the SEC (13).

A.2.1 Cycle-cover inequalities

The cycle-cover inequalities exploit the minimum prize constraint and the fact that a feasible solution must be a cycle. Let $S \subset V$, $1 \in S$ such that $p(S) < \bar{p}$, then

$$x(E(S)) \leq y(S) - 1 \tag{26}$$

is a valid inequality for the PCTSP, as shown in [5].

A.2.2 Conditional inequalities

An upper bound c_U on the objective value can be used to derive inequalities similar to the cycle-cover, but based on the selected edges. Although they are not guaranteed to be valid, these inequalities can be conditionally used in a cutting-plane context. Let $T \subseteq E$ such that $c(T) > c_U$, then

$$x(T) \leq y(V(T)) - 1 \tag{27}$$

is valid for the PCTSP if no feasible solution of value lower than c_U is contained in T, since $x(T) \leq y(V(T))$ holds for every feasible solution. This occurs, in particular, when T defines a simple cycle that goes through the depot and for which $c(T) > c_U$.

A.3 Comb inequalities

The well known comb inequalities can be adapted from the TSP to the PCTSP [4]. Let us consider two sets of vertices, the handle $H \subset V$ and the teeth $T_j \subset V$ ($j = 1, .., t$). The general comb inequalities are formulated as:

$$x(E(H)) + \sum_{j=1}^{t} x(E(T_j)) \leq y(H) + \sum_{j=1}^{t} |T_j| - \frac{3t+1}{2} \tag{28}$$

for all H, T_1, ..., T_t satisfying:

a) $|T_j \cap H| \geq 1$, with $j = 1, .., t$;
b) $|T_j \setminus H| \geq 1$, with $j = 1, ..., t$;
c) $T_i \cap T_j = \emptyset$, with $1 \leq i < j \leq t$; and
d) $t \geq 3$ and odd.

In the special case where $|T_j \cap H| = 1$ for all j, the inequalities are referred to as *simple comb* inequalities. Simple comb inequalities become *2-matching* inequalities if $|T_j \setminus H| = 1$ for all j.

A.4 Logical inequalities

Obviously, if an edge $e \in \delta(v)$ is part of a solution, the vertex v must be visited, hence the following logical inequality:

$$x_e \leq y_v \quad \forall e \in \delta(v), \quad v \in V' \tag{29}$$