Paving and computing the set of nondominated points for the bi-objective 0/1 uncapacitated facility location problem

Xavier Gandibleux* and Anthony Przybylski†
Nantes Université – France

Abstract

The paper presents a three-phase algorithm to compute the set of nondominated points for the binary version of the uncapacitated facility location problem with two objectives. The first phase constructs a *paving* in objective space which is a collection of boxes that covers all nondominated points. The paving procedure is a branch and bound algorithm where boxes are filtered using three dominance-based pruning tests. The second phase is a *refinement* procedure which attempts to either shrink or eliminate boxes from the paving using supported points calculated in the boxes. The third phase is a *generation* procedure which solves the allocation problem associated to each of the remaining boxes using a labeling algorithm. The points obtained in each box are filtered over all boxes and provide the set of nondominated points of the initial problem. Computational experiments on standard benchmark instances demonstrate speedups of several orders of magnitude over both a generic algorithm based on a ε-constraint method using Gurobi as MIP solver and a specific algorithm proposed in 2003 by Fernandez and Puerto.

Keywords: Multi-objective combinatorial optimization – Uncapacited facility location problem – Allocation problem – Branch-and-bound algorithm – Bound sets – Labeling algorithm.

*Xavier.Gandibleux@univ-nantes.fr
†Anthony.Przybylski@univ-nantes.fr

1 Introduction

A multi-objective optimization problem is defined by

$$\min_{\mathrm{x}\in X}(f^1(\mathrm{x}),\ldots,f^p(\mathrm{x})), \tag{MOP}$$

where f^k with $k \in \{1,\ldots,p\}$ are real-valued objective functions defined on a set $X \subseteq \mathbb{R}^{\mathrm{n}}$.
In this work, we will only consider multi-objective combinatorial optimization (MOCO) problems [8, 10] that are a particular case of multi-objective optimization problems for which the objective functions and the constraints are linear, and the variables are binary. Any $\mathrm{x} \in \mathbb{R}^{\mathrm{n}}$ denotes a solution corresponding to decisions, $\mathbb{R}^{\mathrm{n}}$ is the decision space. X is the feasible set, and if $\mathrm{x} \in X$, x is a feasible solution. By $Y = f(X) := \{f(\mathrm{x}) : \mathrm{x} \in X\} \subseteq \mathbb{R}^p$ we denote the image of the feasible set in the objective space $\mathbb{R}^p$, also called outcome set. Any vector $y \in \mathbb{R}^p$ is called a point, and if there is a feasible solution $\mathrm{x} \in X$ such that $y = f(\mathrm{x})$ then y is called a feasible point.

We assume that no feasible point minimizes all objective functions simultaneously. The usual notations for componentwise orders in $\mathbb{R}^p$ are therefore necessary. For $y^1, y^2 \in \mathbb{R}^p$ we shall use the notation $y^1 \leqq y^2$ if $y^1_k \leqq y^2_k$ for all $k = 1,\ldots,p$; $y^1 \leq y^2$ if $y^1 \leqq y^2$ and $y^1 \neq y^2$; and $y^1 < y^2$ if $y^1_k < y^2_k$ for all $k = 1,\ldots,p$. $\mathbb{R}^p_{\leqq}$ denotes the non-negative orthant $\{y \in \mathbb{R}^p : y \geqq 0\}$, $\mathbb{R}^p_{\geq}$ and $\mathbb{R}^p_{>}$ are defined analogously. We consider efficient solutions of (MOP), i.e., a feasible solution $\mathrm{x} \in X$ is called efficient if there does not exist x' $\in X$ such that $f(\mathrm{x}') \leq f(\mathrm{x})$. In other words, no feasible solution is at least as good as x for all objectives, and strictly better for at least one. If x is efficient then $f(\mathrm{x}) = (f^1(\mathrm{x}),\ldots,f^p(\mathrm{x}))$ is said nondominated. The set of all nondominated points is denoted by Y_N and can alternatively be defined by $Y_N := \{y \in Y : \nexists y' \in Y \text{ such that } y' \leq y\}$. The same way, locally nondominated points relatively to any set $S \subset \mathbb{R}^p$ can be defined by $S_N := \{y \in S : \nexists y' \in S \text{ such that } y' \leq y\}$.

In absence of further precision, the exact solution of a multi-objective optimization problem is understood as computing a complete set of efficient solutions X_E, i.e. a set of efficient solutions such that $Y_N = f(X_E)$. Contrary to Y_N, a complete set of efficient solutions is generally not unique. This is due to the existence of equivalent solutions, that are different feasible solutions whose image corresponds to the same nondominated point. A minimal complete set X_{E_m} is a complete set without equivalent solutions. Any complete set contains a minimal complete set.

A distinction is generally done between the efficient solutions of (MOCO). Given $\lambda \in \mathbb{R}^p_{>}$, a solution of the weighted sum scalarization

$$\min_{\mathrm{x}\in X} \sum_{k=1}^{p} \lambda_k f^k(\mathrm{x}) \tag{1}$$

is efficient [16]. However, every efficient solutions is not an optimal solution of a weighted sum scalarization. Efficient solutions that are optimal for some weighted sum problems (1) are called supported efficient solutions. Other efficient solutions are called non-supported efficient solutions and are practically harder to compute. Generally, enumerative methods are necessary to determine the non-supported efficient solutions. In order to reduce the necessary enumeration, the notion of *bound set* is crucial [9].

Sets of points in $\mathbb{R}^p$ called bound sets are used to bound S_N, where S is generally either Y or the feasible set of a subproblem. A complete study about the bound sets can be found in Ehrgott and Gandibleux [11]. In the following, we will use a simple definition of bound set. A lower bound set L for S_N is a set L such that no point in L dominates another one, and for all $y \in S$, there is $l \in L$ such that $l \leqq y$. An upper bound set U for S_N is a set U such that no point of U dominates another one, and defined by points in S.

A well-known particular case of lower bound set for S_N defined by a single point is the ideal point defined by

$$y_k^I := \min_{y \in S} y_k; \quad k = 1, \ldots, p.$$

Our work will be next restricted to the bi-objective case. In this context, a possible way to compute the ideal point is given by $y^I = (y_1^{12}, y_2^{21})$ where y^{12} and y^{21} are lexicographic optimal points over S respectively for the permutation of objectives (12) and (21). The nadir point defined by

$$y_k^N := \max_{y \in S_N} y_k; \quad k = 1, \ldots, p$$

can also be deduced in the bi-objective case using y^{12} and y^{21} by $y^N = (y_2^{12}, y_1^{21})$. y^N does not necessarily belong to S and is therefore not an upper bound set following the simple definition introduced above. Such an upper bound set for S_N is simply given by $\{y^{12}, y^{21}\}$. Finally, all the locally nondominated points relatively to S are located in the rectangle whose extreme points are y^{12}, y^I, y^{21}, y^N.

Most of the single-objective combinatorial optimization problems are NP-hard and consequently their multi-objective variant is also NP-hard. The complexity class of most of the multi-objective combinatorial optimization problems whose single-objective variant is in class P, is still open [3]. Independently of their complexity class, MOCO problems are intractable, i.e., there may exist exponentially many nondominated points and efficient solutions.

2 Problem Formulation and related literature

The bi-objective 0/1 uncapacitated facility location problem (2-UFLP) generalizes the well-known NP-hard single-objective UFLP [5, 20] to the case of two cost objectives. It arises naturally in practical settings such as telecommunication network design [17]. Eiselt and Laporte [12] present a panorama of the objective functions commonly met in the problems of localization of services. Taking into consideration this panorama, it is not surprising to notice the growing interest in facility location problems where several objectives have to be optimized simultaneously (see e.g. [8, 10, 13, 21]).

2.1 The bi-objective 0/1 uncapacitated facility location problem

The 2-UFLP considered in this paper is a discrete problem which can be stated as follow. Suppose there are m customers, n potential facilities, and two objective functions. Let $I = \{1, \ldots, m\}$ be the set of customer indices and $J = \{1, \ldots, n\}$ the set of potential facility indices. For $k \in \{1, 2\}$, $c_{ij}^k \geq 0$ denotes the assignment cost of customer i to facility j under objective k, and $r_j^k \geq 0$ the opening cost of facility j under objective k. The decision variables are $x_{ij} \in \{0, 1\}$ where x_{ij} is equal to 1 if the whole demand of the customer i is provided by the facility j (0 otherwise), and $s_j \in \{0, 1\}$ where s_j is equal to 1 if the facility j is open (0 otherwise). The problem consists in opening a selection of facilities and assigning the customers to these facilities, while minimizing two linear objective functions $f^k(x, s)$. A formulation of the 2-UFLP is given below.

$$\left[\begin{array}{lll}
\text{Min} & \left\{ f^k(x,s) = \displaystyle\sum_{i \in I} \sum_{j \in J} c_{ij}^k x_{ij} + \sum_{j \in J} r_j^k s_j \qquad k = 1, 2 \right\} & (0) \\
\text{subject to} & \displaystyle\sum_{j \in J} x_{ij} = 1 \qquad \forall i \in I & (1) \\
& x_{ij} \leq s_j \qquad \forall i \in I,\ \forall j \in J & (2) \\
& x_{ij},\ s_j \in \{0, 1\} \qquad \forall i \in I,\ \forall j \in J & (3)
\end{array}\right]$$

The constraint (1) expresses the fact that the demand of customer i is supplied by one facility, and (2) expresses the fact that a customer can be assigned only to an open facility. The objective

functions to minimize are composed of the assignment cost and of the cost of opening the facilities, called operating cost.

To the best of our knowledge, only Fernández and Puerto have proposed in [14] a specific algorithm for calculating the set of nondominated points for the 2-UFLP. Their approach is based on a dynamic programming algorithm. Numerical instances have been solved with up to 20 facilities and 50 customers.

There are more studies aimed at proposing an approximation of the set of all nondominated points for the 2-UFLP using algorithms based on metaheuristics. Although this is beyond the scope of this paper, we should nevertheless mention the work of Harris et al. [18, 19] which belongs to the context of green logistics, given that its numerical instances are considered in our experiments.

2.2 Illustrative example

We consider the following illustrative example $|J|$=5 facilities and $|I|$=8 customers. The costs are uniformly generated in the set $\{1,\ldots,100\}$. The fixed costs are given by r^1=(93 162 156 724 92) and r^2=(92 214 330 468 414). The assignment costs are given by the following matrices:

$$c^1 = \begin{pmatrix} 7 & 20 & 41 & 13 & 34 \\ 24 & 71 & 28 & 11 & 25 \\ 69 & 88 & 37 & 10 & 57 \\ 76 & 64 & 49 & 11 & 29 \\ 76 & 67 & 47 & 15 & 76 \\ 8 & 44 & 26 & 12 & 27 \\ 69 & 58 & 36 & 15 & 20 \\ 96 & 54 & 38 & 16 & 13 \end{pmatrix} \text{ and } c^2 = \begin{pmatrix} 33 & 99 & 45 & 11 & 71 \\ 26 & 31 & 10 & 10 & 97 \\ 70 & 15 & 87 & 12 & 110 \\ 73 & 66 & 29 & 11 & 72 \\ 2 & 93 & 48 & 10 & 68 \\ 44 & 22 & 50 & 18 & 45 \\ 35 & 51 & 6 & 17 & 95 \\ 55 & 37 & 73 & 17 & 74 \end{pmatrix}$$

Several mathematical programming software that natively support features for modeling and solving multi-objective optimization problems have recently appeared [15]. Thus, the use of the JuMP [7] and MultiObjectiveAlgorithms [6] packages –which are, respectively, an algebraic modeling language and a meta-solver incorporating various algorithms from the literature and utilizing an MIP solver, both available in the Julia programming language– enables the description and the resolution of a 2-UFLP instance. Listing 1∼3 report Julia codes of the didactic instance using these two packages, and Gurobi as MIP solver.

```
C = cat( [  7 20 41 13 34;  24 71 28 11 25;  69 88 37 10  57;  76 64 49 11 29;
           76 67 47 15 76;   8 44 26 12 27;  69 58 36 15  20;  96 54 38 16 13 ],
         [ 33 99 45 11 71;  26 31 10 10 97;  70 15 87 12 110;  73 66 29 11 72;
            2 93 48 10 68;  44 22 50 18 45;  35 51  6 17  95;  55 37 73 17 74 ],
         dims=3)

R = cat( [93, 162, 156, 724, 92],
         [92, 214, 330, 468, 414],
         dims=2)
```

Listing 1: Encoding in Julia the data of the didactic problem.

```
using JuMP, Gurobi
import MultiObjectiveAlgorithms as MOA

moo = Model(() -> MOA.Optimizer(Gurobi.Optimizer))
set_attribute(moo, MOA.Algorithm(), MOA.EpsilonConstraint())

I = 1:size(C,1)
J = 1:size(C,2)
@variable(moo, x[i∈I,j∈J], Bin)
@variable(moo, s[j∈J], Bin)
@expression(moo, f[k=1:2], sum(C[i,j,k]x[i,j] for i∈I,j∈J)+sum(R[j,k]s[j] for j∈J))
```

```
@objective(moo, Min, [f[k] for k=1:2])
@constraint(moo, [i∈I], sum(x[i,j] for j∈J) == 1)
@constraint(moo, [i∈I,j∈J], x[i,j] <= s[j])

optimize!(moo)
```

Listing 2: Formulation and resolution of a 2-UFLP with JuMP and the ϵ-constraint algorithm provided by MultiObjectiveAlgorithms. Here, Gurobi is selected as MIP solver.

```
for r in 1:result_count(moo)
    println("Solution␣$r")
    println("␣␣␣x␣=␣", [(i,j) for j∈J,i∈I if value(x[i,j]; result = r) > 0.5])
    println("␣␣␣s␣=␣", [(j) for j∈J if value(s[j]; result = r) > 0.5])
    println("␣obj␣=␣", objective_value(moo; result = r))
end
```

Listing 3: Collect and display the results.

Listing 1 shows the data encoding of the didactic problem: a three-dimensional cost matrix `C` (8 customers x 5 facilities x 2 objectives) and a fixed-cost matrix `R` (5 facilities x 2 objectives). In listing 2, the 2-UFLP is modeled and solved. First, packages to use are declared at lines 1∼2. At lines 4∼5, a JuMP model named `moo` is then declared using the ϵ-constraint algorithm provided by the MultiObjectiveAlgorithms metasolver and Gurobi as MIP solver. Values for sets `I` and `J` are derived from the data at lines 7∼8. Next, the model is described. Lines 9∼10 declare binary assignment variables `x[i,j]` and facility-opening variables `s[j]`. Lines 11∼12 construct the two objective expressions `f[k]` as the total assignment costs plus fixed costs. Lines 13∼14 ensures that each customer is assigned to exactly one open facility. Finally, the instance is solved calling `optimize!` at line 16. Listing 3 query and display the optimization results. A iterator over all solutions returned by the solver is performed, and values are printed for each the active assignments, the open facilities, and the corresponding bi-objective value. Table 1 reports Y_N and X_{E_m} collected at the end of the resolution.

Y_N	X_{E_m}	
y	$s_j = 1$	$x_{ij} = 1$
(373, 1046)	$j \in \{5\}$	$(i,j) \in \{(1,5)(2,5)(3,5)(4,5)(5,5)(6,5)(7,5)(8,5)\}$
(419, 962)	$j \in \{1,5\}$	$(i,j) \in \{(1,1)(2,1)(3,5)(4,5)(5,1)(6,1)(7,5)(8,5)\}$
(431, 922)	$j \in \{1,5\}$	$(i,j) \in \{(1,1),(2,1),(3,1),(4,5),(5,1),(6,1),(7,5),(8,5)\}$
(458, 678)	$j \in \{3\}$	$(i,j) \in \{(1,3)(2,3)(3,3)(4,3)(5,3)(6,3)(7,3)(8,3)\}$
(518, 430)	$j \in \{1\}$	$(i,j) \in \{(1,1)(2,1)(3,1)(4,1)(5,1)(6,1)(7,1)(8,1)\}$

Table 1: Y_N and X_{E_m} for the didactic instance.

2.3 The bi-objective allocation problem

An important sub-problem of the facility location problem without capacity constraint is the allocation problem. This problem appears once a set of facilities $J_1 \subseteq J$ has been selected. The problem consists of assigning customers to the facilities. In the two-objective case, this leads to

the following formulation.

$$\left[\begin{array}{lll} \text{Min} & \left\{ f^k(x) = \sum_{i\in I}\sum_{j\in J_1} c_{ij}^k x_{ij} + \sum_{j\in J_1} r_j^k \qquad k = 1,2 \right\} & (0') \\ s/c & \sum_{j\in J_1} x_{ij} = 1 \qquad \forall i \in I & (1) \\ & x_{ij} = 0 \qquad \forall i \in I,\ \forall j \in J \setminus J_1 & (4) \\ & x_{ij} \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J_1 & (3') \end{array}\right]$$

which is simply formulation 2-UFLP in which variables s_j for $j \in J_1$ are set to 1, and s_j for $j \in J \setminus J_1$ are set to 0.

This problem is straightforward in the single-objective case, as it consists just in assigning each customer independently to the open facility with the smallest cost. In the bi-objective case, the problem has been studied in [14] and [23]. Fernández and Puerto [14] have proposed to reformulate the problem as a shortest path problem. They have also proposed a parametric algorithm to determine the set of supported nondominated points of this problem. Stiglmayr et al. [23] have conducted a theoretical study on this problem, and in particular demonstrated that it is intractable and that its associated decision problem is NP-complete. They have also proposed a reformulation as a bi-objective multiple choice knapsack problem, that is next reformulated as a shortest path problem. Next, Stiglmayr et al. [23] have been more specific about the solution method that is applied. They have in particular compared two strategies for the dominance filtering of labels in the solution of the obtained shortest path problem.

2.4 Algorithm in Three-Phase proposed

This paper presents an original three-phase algorithm (Algorithm 1) to compute the set of nondominated points Y_N for the 2-UFLP.

Algorithm 1 `biUFLPsolver`

input: *data* ▷ numerical instance
output: Y_N ▷ nondominated points
1: $initialPaving \leftarrow$ `computePaving`($data$) ▷ phase 1
2: $improvedPaving \leftarrow$ `improvePaving`($data$, $initialPaving$) ▷ phase 2
3: $Y_N \leftarrow$ `generateNDPoints`($data$, $improvedPaving$) ▷ phase 3

During Phase 1 (line 1), an initial paving of the nondominated set Y_N –a collection of boxes that covers all nondominated points– is computed. It is composed only of boxes obtained by lexicographic optimal points of the associated bi-objective allocation problem. Each box is potentially candidate for generating a subset of Y_N. This paving is obtained using a branch and bound algorithm. During Phase 2 (line 2), the initial paving is refined according to a principle aimed at shrinking and pruning the boxes. For this purpose, Phase 2 enumerates some additional feasible points in each box and consequently, additional pruning tests can be applied. During Phase 3 (line 3), locally nondominated points are computed in each box of the paving by the solution of an allocation subproblem, and are next merged to obtain the set Y_N of 2-UFLP.

The remaining of the paper is organized as follows. Sections 3, 4, and 5 present, respectively, phases 1, 2, and 3 of the proposed algorithm. Finally, Section 6 reports and analyzes computational experiments performed on standard benchmark instances.

3 Phase 1: the paving

3.1 Definitions and notations

Any feasible solution for the 2-UFLP problem involves one or more facilities that must be opened, and assigns all customers to the open facilities. Given this, let $J_1 \subseteq J$ denote a set of indices of opened facilities for a given feasible solution, with (1) $\mid J_1 \mid \geq 1$, (2) $s_j = 1$ for all $j \in J_1$ ($s_j = 0 \; \forall j \in J \setminus J_1$), and (3) customers have to be assigned to open facilities $j \in J_1$.

Definition 1. *Let $J_1 \subseteq J$, the point corresponding to the* operating cost vector $y^R \in \mathbb{R}^2$ *resulting from the opening of these facilities is immediately obtained:*

$$y_k^R = \sum_{j \in J_1} r_j^k, \quad k = 1, 2. \tag{2}$$

The definition of a box associated with a set J_1 is now introduced.

Definition 2. *A* box $\mathcal{B}_{J_1}$ *associated with J_1 is a 2-dimensional subset of the objective space $\mathbb{R}^2$ bounded by (at most) two* lexicographic optimal points y^{12} *and y^{21} for the allocation sub-problem defined by J_1.*

To compute the lexicographic optimal solutions of a bi-objective allocation problem is simply done by assigning the customers to the open facility that has the lexicographic smallest cost. The image of this solution is obtained by summing the operating cost vector y^R and the assignment costs.
y^{12} and y^{21} are two remarkable points of a box from which several observations are made. First these two points are nondominated for the considered allocation problem, and they are two potential nondominated points for the 2-UFLP. It is also worth noting that y^{12} and y^{21} are points corresponding to feasible solutions, which may be used in the branch-and-bound for pruning nodes in the tree. Obviously, all nondominated points of the 2-ULFP problem for which the set of selected facilities is J_1 lie within the box. As every feasible solution of the 2-UFLP are associated to a set J_1 of open facilities, none of the nondominated points of the 2-UFLP can exist outside a box. Second, two other remarkable points are also associated with a box $\mathcal{B}_{J_1}$, the ideal point y^I, and the nadir point y^N. Knowing the lexicographic optimal points, the ideal point and the nadir point can be immediately deduced in the bi-objective case. Finally, when $|J_1| = 1$, all remarkable points coincide.

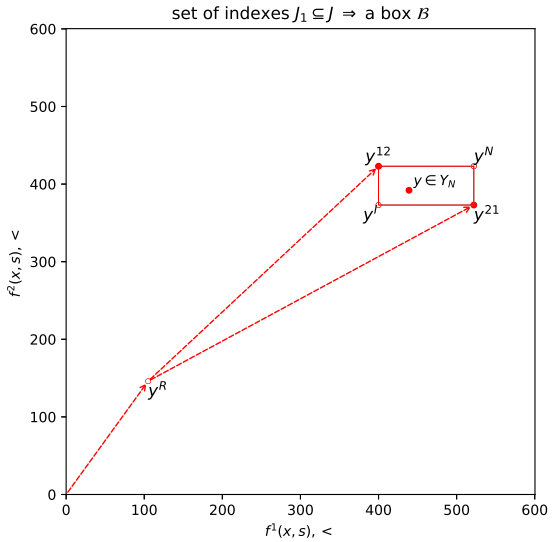


Figure 1: A box is then a rectangle in objective space, whose edges are parallel to the axes, with four remarkable points: y^{12}, y^{21}, y^I, y^N. The point $y \in Y_N$ illustrates one nondominated point belonging to this box for J_1.

For the sake of simplicity, and since there is no possibility of confusion, the set J_1 will no longer be mentioned in the remainder of this paper when referring to a box $\mathcal{B}$. The notion of paving is now formally defined.

Definition 3. *A* paving $\mathcal{P}$ *of the nondominated set* Y_N *is a collection of boxes such that* $Y_N \subset \bigcup_{\mathcal{B}\in\mathcal{P}} \mathcal{B}$.

A paving $\mathcal{P}$ provides a way to bound the nondominated set Y_N. It is therefore natural to question the interest of the notion of paving with respect to the notion of bound set for Y_N. From a paving, it is indeed possible to deduce a pair of bound sets for Y_N. With y^I, the ideal point associated to a box $\mathcal{B}$, then $(\bigcup_{\mathcal{B}\in\mathcal{P}} y^I)_N$ defines a lower bound set for Y_N. With y^{12} and y^{21}, the feasible points defining the box $\mathcal{B}$, then $(\bigcup_{\mathcal{B}\in\mathcal{P}}\{y^{12}\} \cup \{y^{21}\})_N$ defines an upper bound set for Y_N. Hence, from the point of view of bound sets, a paving contains redundant information that must be filtered. However, the use of a paving will reveal to be interesting for the design of a solution method as each box is associated to an information (associated open facilities) that is ignored with the use of bound sets.

The paving algorithm can now be described. As $J = \{1, \ldots, n\}$, there are potentially $2^n - 1$ boxes to consider, i.e. the open facilities can be any subset of J such that $|J| \geq 1$. The aim of the paving algorithm is to determine a set $\mathcal{B}$ of boxes that defines a paving of Y_N using the least possible of boxes.

3.2 Didactic example (continued)

All sets J_1 for this problem are given by Figure 2. If none of these combinations of facilities may be pruned, the full search tree over subsets $J_1 \subseteq J$ has $2^5 - 1 = 31$ non-empty nodes. It corresponds exactly to Figure 2.

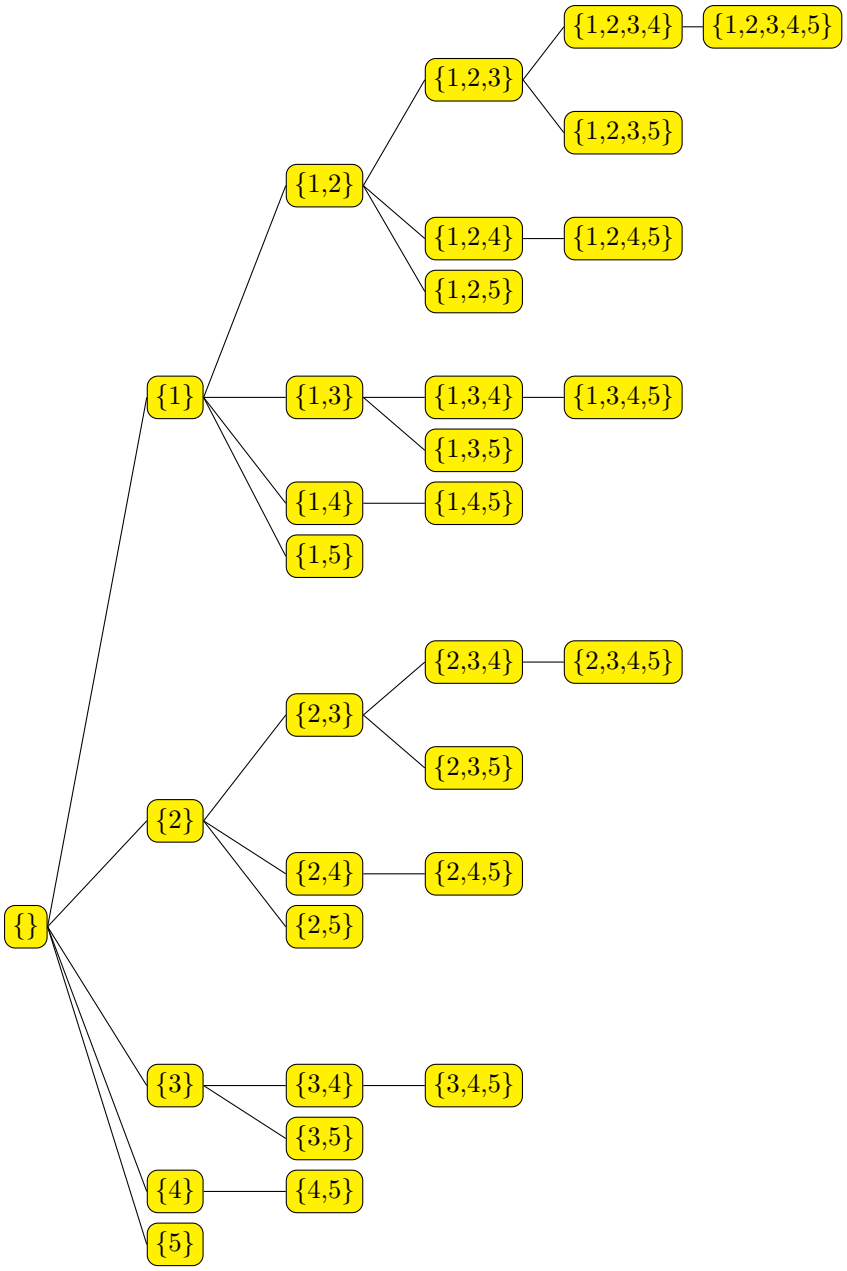


Figure 2: All sets J_1 for the illustrative example.

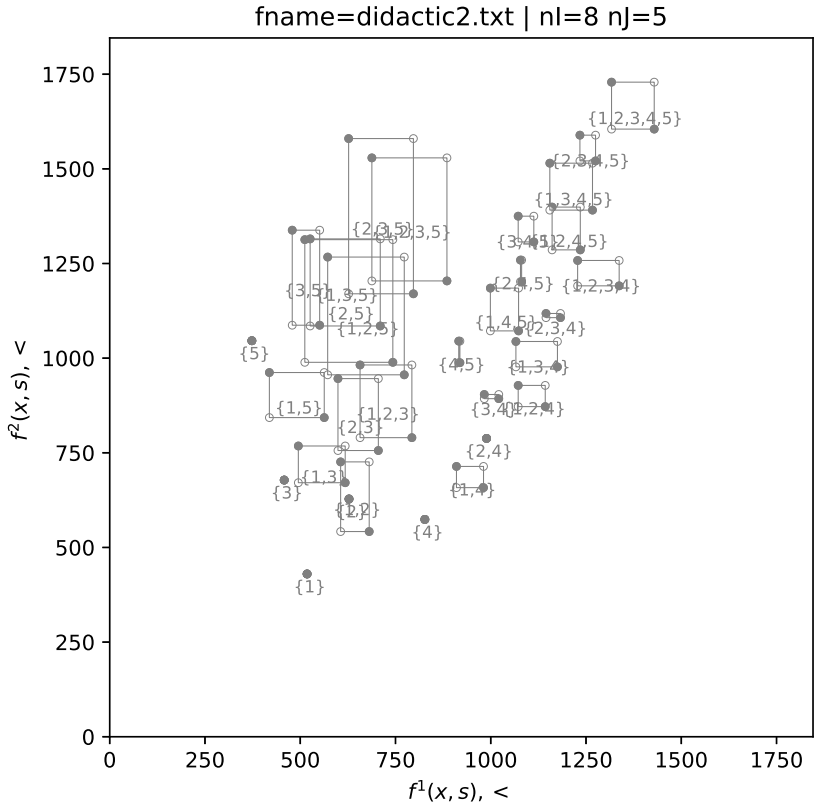


Figure 3: Paving obtained without performing prunings.

The corresponding paving obtained without pruning tests is depicted by Figure 3.

3.3 Initial paving with a branch-and-bound algorithm

An initial paving is built using an unconventional branch and bound algorithm. The principle is indeed particular, as the purpose is not to return the set Y_N of nondominated points but to return a paving $\mathcal{P}$.

The construction of the search tree is initially classical. Each node of the branch and bound tree is defined by a subset of open facilities, a subset of closed facilities, and a subset of unset facilities. In other words, each node is defined by a vector of values for the variables s_j, some of which may be unfixed. More precisely, given a node with a subset of open facilities J_1=$\{j_1, \ldots, j_l\}$ (that we suppose ordered by increasing index), the facilities indexed by $\{1, \ldots, j_l\} \setminus J_1$ will necessarily be closed. For example, for a problem with 5 facilities, if $J_1 = \{1, 3\}$ then the facilities 1 and 3 are open, the facility 2 is closed, the facilities 4 and 5 are unset. The search tree defined by these nodes is explored following a breadth-first search strategy. New nodes are generated by opening an additional facility. Suppose that a node is defined by a subset J_1 of open facilities where $J_1 = \{j_1, \ldots, j_l\}$, then if no pruning test can be applied, the open facilities of the generated child nodes are: $J_1 \cup \{j_l + 1\}, \ldots, J_1 \cup \{n\}$.

In such a search tree, the variables s_j are all fixed only for leaf nodes. An allocation subproblem is therefore set only for these nodes. To reach the leaf nodes is necessary to generate boxes and this would happen late with a breadth-first search exploration. An additional mechanism called *expansion* of a node is added in the exploration of this search tree. When new nodes are generated from the node defined by a set J_1 of open facilities, we immediately generate the

leaf node with the same set J_1 of open facilities, the facilities in $J \setminus J_1$ are therefore closed. It becomes possible to define the corresponding box, by computing lexicographic optimal points of the corresponding bi-objective allocation problem. Feasible points for 2-UFLP are obtained with the computation of each box, and it happens from the early steps of the enumeration. Next, feasible points and in particular bound sets will be used to prune nodes of the search tree.

The cost vector of any feasible solution of 2-UFLP is composed of the operating cost vector y^R (Definition 1) and the cost of assigning the customers to the open facilities. Consequently, y^R defines an immediate lower bound set for the nondominated set of any subproblem for which J_1 is a subset of the open facilities. This is clearly a very weak bound set but it comes with a very low computational cost. Note: This lower bound set was also used in [14].

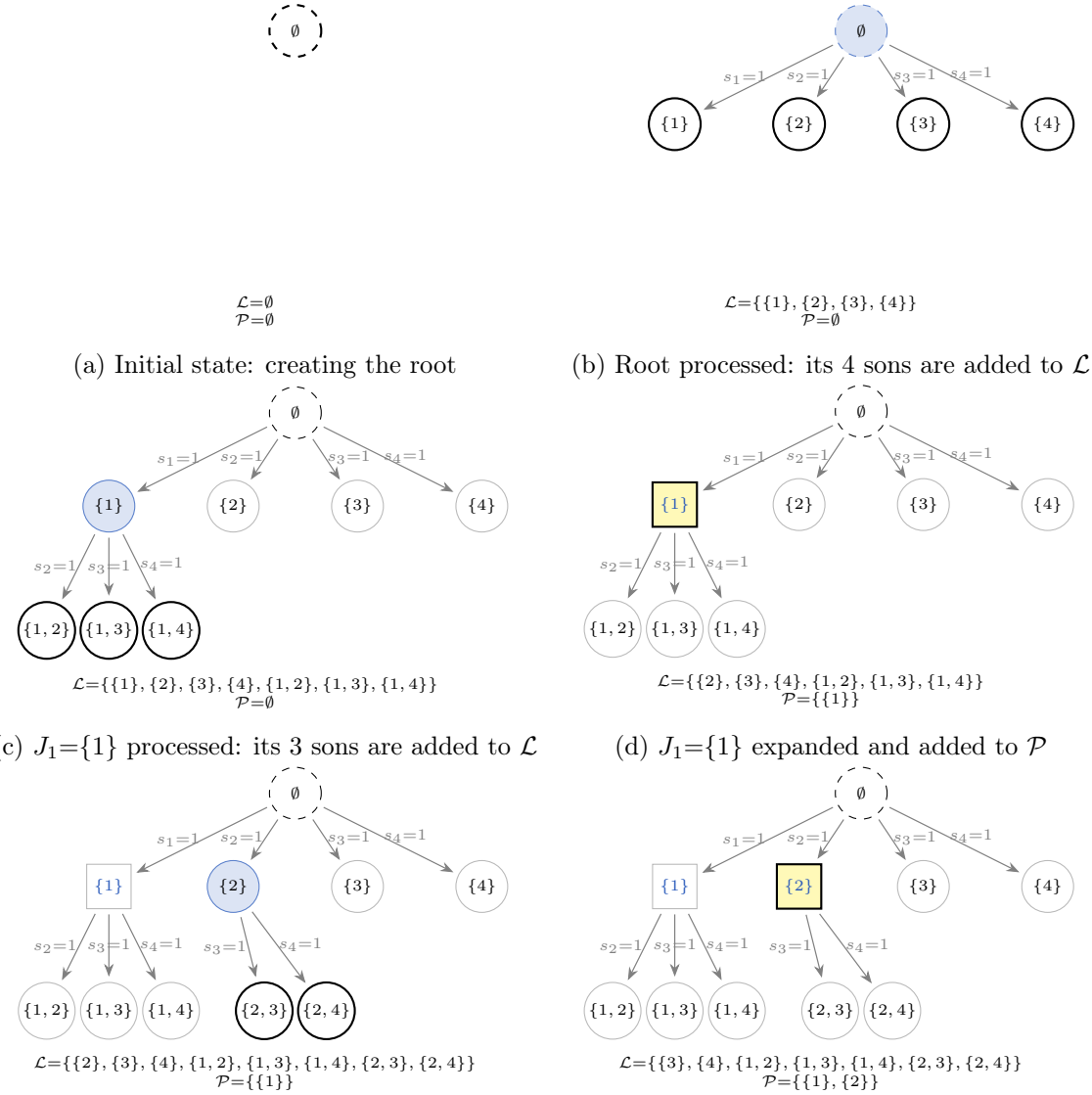
Next, after the expansion of a node, two bound sets can be deduced from the associated box $\mathcal{B}$. The ideal point y^I of the box $\mathcal{B}$ is an immediate lower bound set for the nondominated set of the corresponding allocation problem.

Algorithm 2 summarizes the branch and bound algorithm. Two sets of nodes are managed: the set $\mathcal{L}$ of unpruned (unexpanded) nodes, and the set $\mathcal{P}$ of unpruned expanded nodes. Both kinds of nodes are described by a set of open facilities. As the search tree will be explored following a breadth-first search strategy, the sets $\mathcal{L}$ and $\mathcal{P}$ are naturally implemented by queues. In Algorithm 2, the nodes are denoted by upper-case letters: N for a unexpanded node, and B for an expanded node whose associated box is $\mathcal{B}$. In a conventional branch and bound algorithm, leaf nodes correspond to subproblems that are completely solved and are therefore pruned. Here, we do not completely solve the allocation problems as only their lexicographic points are computed. The purpose of this branch and bound algorithm is indeed not to obtain a set Y_N but to obtain a paving that contains only potentially useful boxes, i.e. only boxes for which we can prove that they do not contain any nondominated point of 2-UFLP will be pruned. The output of the algorithm will therefore be the set of unpruned nodes, each of which define an allocation subproblem with an associated box.

Two procedures are used to apply the branching strategy. The procedure `branch` takes in parameter a unexpanded node, and returns the set of unexpanded nodes opening one additional facility. The procedure `expand` takes in parameter a unexpanded node, and returns the node expanded. The corresponding box is immediately determined and the upper bound set U for Y_N may be updated. Figure 4 illustrates these different operations on a problem involving four facilities. After creating the root (Figure 4a), the set $\mathcal{L}$ is initialized with the unexpanded nodes defined by only one open facility (Figure 4b). There is initially no expanded node and no known feasible point, and thus $\mathcal{P}$ and U are initialized with the empty set (Line 1-3 of Algorithm 2). Figures 4c∼4f illustrates the sequence of operations branch/expand along the search tree -when pruning test are disabled- for some iterations.

The main loop (line 4-18 of Algorithm 2) can now start (See Figures 5∼6 for an illustration of the operations using the 4-facility example, where the current node N with value {3} is processed). While the queue $\mathcal{L}$ is not empty, the first node N in $\mathcal{L}$ is extracted from the queue (Figure 5a) and a first pruning test is applied to check if it is necessary to branch on this node (Figure 5b). This is done by the use of the predicate `Test1` which tests if the lower bound set y^R for the nondominated set of the subproblem defined by the node N is weakly dominated by a known feasible point. If this is the case, the node N is pruned and the iteration of the while loop ends (Figure 5c). If the first pruning test cannot be applied, we branch on the node N, which implies an update of the queues $\mathcal{L}$ (line 8-9 of Algorithm 2) (Figure 5d). Next, we expand the node N to obtain the leaf node B. This implies an update of the queue $\mathcal{P}$, and also a possible update of the upper bound set U (line 10-11 of Algorithm 2) (Figure 5e). Finally, it is tested if the expanded node B can be fathomed or not, in other words if its associated box may be useful in the definition of a paving of Y_N. It is done using the predicate `Test2` which tests if the ideal point of the box associated to the node B is weakly dominated by a feasible point (Figure 6a). If this is the case, the new expanded node B can be immediately deleted from the queue $\mathcal{P}$

(Figure 6b)(Line 13-13 of Algorithm 2). Note that there is no need to apply `Test2` if U has been updated with the creation of B since this test would necessarily be false. Finally, if U has been updated, the procedure `Test3` checks if the new points of U can be used to fathom expanded nodes (Figure 6c∼6d) (Line 17 of Algorithm 2). It is not tested if these points can be used to fathom unexpanded nodes, as `Test1` checks this when a unexpanded node is selected.



(a) Initial state: creating the root

(b) Root processed: its 4 sons are added to $\mathcal{L}$

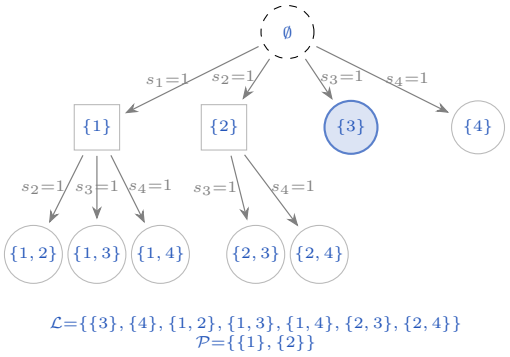
(c) J_1={1} processed: its 3 sons are added to $\mathcal{L}$

(d) J_1={1} expanded and added to $\mathcal{P}$

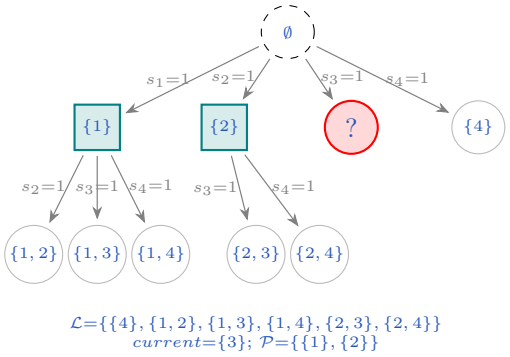
(e) J_1={2} processed: its 2 sons are added to $\mathcal{L}$

(f) J_1={2} expanded and added to $\mathcal{P}$

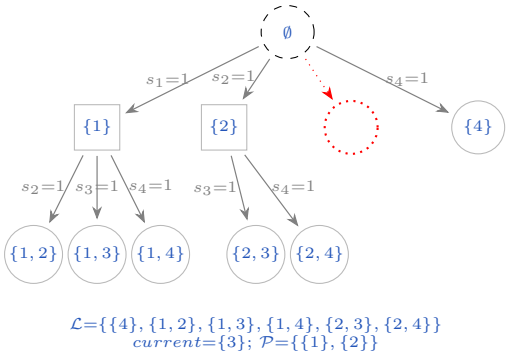
Figure 4: Illustration of the branch-and-bound algorithm using an example with 4 facilities, without activating the pruning tests. The algorithm constructs a paving by systematically exploring all non-empty subsets of services $J_1 \subseteq J$, organised in the form of a decision tree traversed in breadth-first order. Each node of the tree corresponds to a subset J_1. Two lists are used to control the exploration: $\mathcal{L}$ contains the unexpanded nodes (circle) awaiting processing, and $\mathcal{P}$ contains the expanded nodes (square) that constitute the current paving. The figure indicates the correponding sets J_1, values taken by the s_j variables, values taken by queues $\mathcal{L}$ and $\mathcal{P}$ along the iterations of the algorithm.



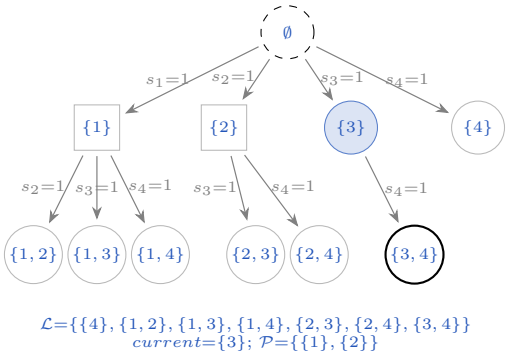
(a) Node J_1={3} is the next node selected in $\mathcal{L}$



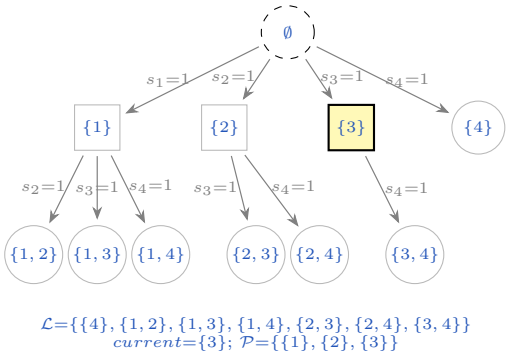
(b) Running `Test1` on node J_1={3}



(c) `Test1`=true ⇒ node J_1={3} pruned

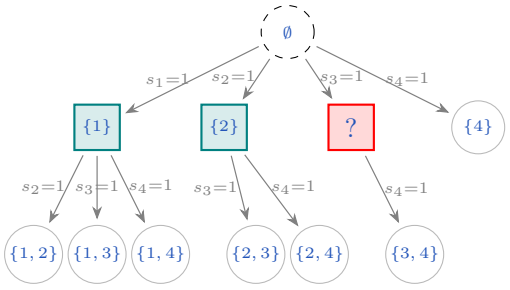


(d) `Test1`=false ⇒ Branch: node J_1={3, 4} added to $\mathcal{L}$



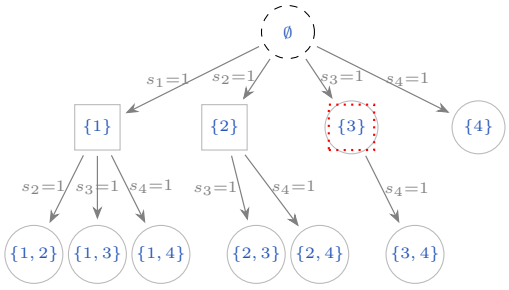
(e) `Test1`=false ⇒ Expand: box J_1={3} added to $\mathcal{P}$

Figure 5: Illustration of the use of pruning tests during the branch-and-bound algorithm (part 1). Let the current situation be given by $\mathcal{L} = \{\{3\}, \{4\}, \{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}\}$ and $\mathcal{P} = \{\{1\}, \{2\}\}$. The node J_1={3} is the current node selected to perform the three tests. The operations following the conclusion of the use of `Test1` are illustrated.



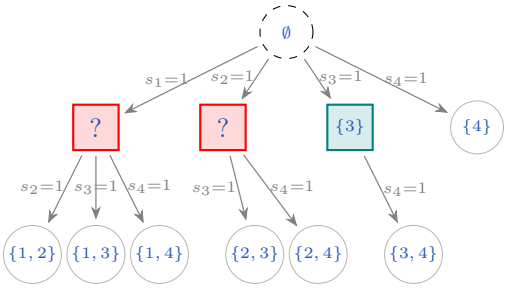
$\mathcal{L}$={{4}, {1, 2}, {1, 3}, {1, 4}, {2, 3}, {2, 4}, {3, 4}}
current={3}; $\mathcal{P}$={{1}, {2}, {3}}

(a) Running `Test2` on box J_1={3}



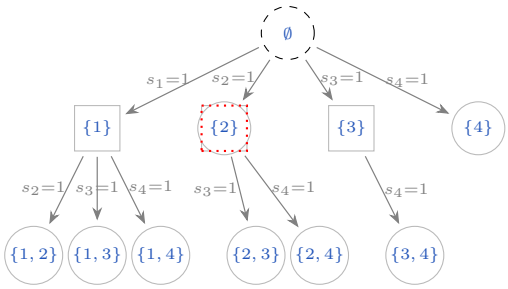
$\mathcal{L}$={{4}, {1, 2}, {1, 3}, {1, 4}, {2, 3}, {2, 4}, {3, 4}}
current={3}; $\mathcal{P}$={{1}, {2}}

(b) `Test2`=true ⇒ box J_1={3} pruned



$\mathcal{L}$={{4}, {1, 2}, {1, 3}, {1, 4}, {2, 3}, {2, 4}, {3, 4}}
current={3}; $\mathcal{P}$={{1}, {2}, {3}}

(c) `Test2`=false ⇒ Running `Test3` using new feasible points associated to J_1={3}



$\mathcal{L}$={{4}, {1, 2}, {1, 3}, {1, 4}, {2, 3}, {2, 4}, {3, 4}}
current={3}; $\mathcal{P}$={{1}, {3}}

(d) `Test3`=true ⇒ box J_1={2} pruned

Figure 6: Illustration of the use of pruning tests during the branch-and-bound algorithm (part 2). The operations following the use of `Test2` and `Test3` are illustrated.

Algorithm 2 PAVING WITH A BRANCH AND BOUND

input: 2-UFLP instance
output: $\mathcal{P}$: a paving of Y_N
$\mathcal{L} \leftarrow \emptyset$; $\mathcal{P} \leftarrow \emptyset$; $U \leftarrow \emptyset$
//Generate new unexpanded nodes from the root node to initialize $\mathcal{L}$
branch($\emptyset \downarrow, \mathcal{N} \uparrow$)
enqueue($\mathcal{N} \downarrow, \mathcal{L} \updownarrow$)
while $\mathcal{L} \neq \emptyset$ **do**
  //Dequeue the node N from $\mathcal{L}$ to be processed
  dequeue($\mathcal{L} \updownarrow, N \uparrow$)
  //Test 1:
  //check if the lower bound set y^R is weakly dominated
  if Test1($N \downarrow, U \downarrow$) = TRUE **then**
    //N is fathomed, nothing to do here
  else
    //Generate new unexpanded nodes from N and update $\mathcal{L}$
    branch($N \downarrow, \mathcal{N} \uparrow$)
    enqueue($\mathcal{N} \downarrow, \mathcal{L} \updownarrow$)
    //Expand N and update $\mathcal{P}$ and U
    expand($N \downarrow, U \updownarrow, B \uparrow$)
    enqueue($B \downarrow, \mathcal{P} \updownarrow$)
    if U has not updated with the creation of B **then**
      //Test 2:
      //check is the ideal point of the expanded node is weakly dominated
      if Test2($B \downarrow, U \downarrow$) = TRUE **then**
        //B is fathomed; it is removed from $\mathcal{P}$
        deleteTail($\mathcal{P} \updownarrow$)
      end if
    else
      //Test 3:
      //check is the ideal point of any expanded node is weakly dominated
      Test3($U \downarrow, \mathcal{P} \updownarrow$)
    end if
  end if
end while

3.4 Didactic example (continued)

Table 2 reports the configurations J_1 generated and the tests triggered for each level of depth in the tree. Figures 7 and 8 illustrates situations where 1 and 2 facilities are respectively opened.

Comments:

- Situations with $\mid J_1 \mid$=1:
  Five configurations are explored.

  1. For $J_1 = \{2\}$: y^I of the expanded node with $J_1 = \{2\}$ is dominated by y^{12} of the expanded node $J_1 = \{1\}$
     ↳ Test2=true ⇒ the expanded node $\{2\}$ pruned
  2. For $J_1 = \{4\}$: y^R of the unexpanded node $J_1 = \{4\}$ is dominated by y^{12} of the expanded node $J_1 = \{1\}$
     ↳ Test1=true ⇒ the unexpanded node $\{4\}$ pruned

Current paving at this step: $\mathcal{P} = \Big\{\{1\},\{3\},\{5\}\Big\}$.

- Situations with $|\ J_1\ |$=2:
  Nine configurations are explored. Among them:

  1. For $J_1 = \{3,5\}$: y^I of the expanded node $J_1 = \{3,5\}$ is dominated by y^{12} of the expanded node $J_1 = \{1,5\}$
     ↳ `Test2`=true ⇒ the expanded node $\{3,5\}$ pruned (see Figure 8).
  2. For $J_1 \in \Big\{\{1,2\},\{1,4\},\{2,3\},\{2,4\},\{2,5\},\{3,4\}\Big\}$: either the unexpanded node is pruned (and therefore the expanded node is not considered) or the expanded node is pruned.

  Current paving at this step: $\mathcal{P} = \Big\{\{1\},\{3\},\{5\},\{1,3\},\{1,5\}\Big\}$.

- Situations with $|\ J_1\ |$>2: all the configurations explored are pruned.

The paving obtained a the end of the branch-and-bound is $\mathcal{P} = \Big\{\{1\},\{3\},\{5\},\{1,3\},\{1,5\}\Big\}$ illustrated by Figure 9. It is easy to notice in the figure that the box defined by $\{1,5\}$ may be refined.

Depth 1		Depth 2		Depth 3		Depth 4	
J_1	test	J_1	test	J_1	test	J_1	test
$\{1\}$		$\{1,2\}$	2	$\{1,2,3\}$	2	$\{1,2,3,4\}$	1
						$\{1,2,3,5\}$	1
				$\{1,2,4\}$	1		
				$\{1,2,5\}$	2		
		$\{1,3\}$		$\{1,3,4\}$	1		
				$\{1,3,5\}$	2		
		$\{1,4\}$	1				
		$\{1,5\}$					
$\{2\}$	2	$\{2,3\}$	2	$\{2,3,4\}$	1		
				$\{2,3,5\}$	2		
		$\{2,4\}$	1				
		$\{2,5\}$	2				
$\{3\}$		$\{3,4\}$	1				
		$\{3,5\}$	2				
$\{4\}$	1						
$\{5\}$							

Table 2: For each level of depth in the tree, configurations J_1 generated and tests triggered. If Test 1 is triggered, the unexpanded node is pruned and it Test 2 is triggered, the expanded node is pruned.

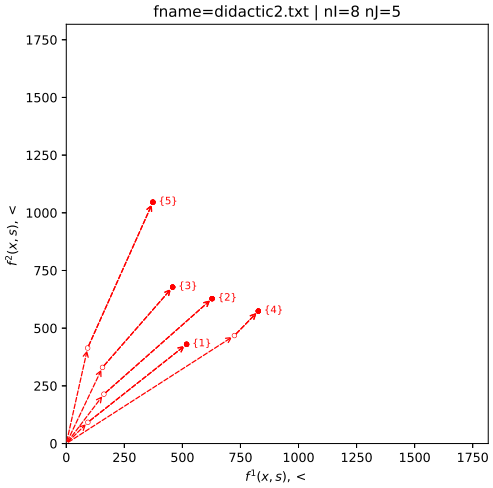


Figure 7: All configurations corresponding to one facility opened.

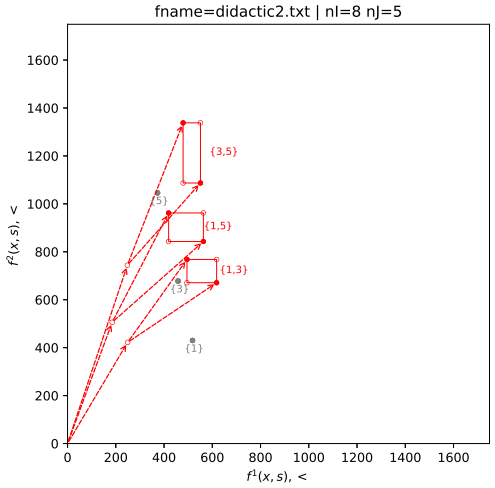


Figure 8: Three configurations corresponding to two facilities opened.

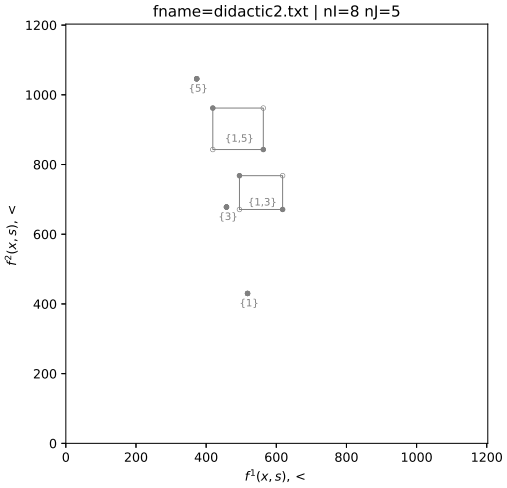


Figure 9: Paving $\mathcal{P} = \Big\{\{1\},\{3\},\{5\},\{1,3\},\{1,5\}\Big\}$ obtained at the end of the branch-and-bound

4 Phase 2: the refinement

4.1 Shrinking and pruning boxes: principle and choices

The paving obtained as output of the branch and bound algorithm may contain a large number of boxes of various size (see Figure 10). It is clear from the figure that, if additional information is available within a box, it is potentially possible either to delete other boxes or to shrink their size.

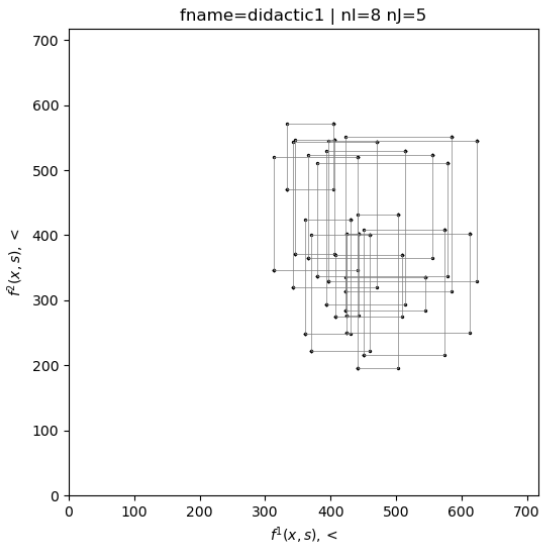


Figure 10: Example of a paving (composed of 17 boxes) obtained at the end of the branch and bound procedure.

However, for a given box —that is, a fixed set J_1— it is straightforward to obtain points that are locally supported nondominated. One simply needs to solve a series of single-objective allocation problems in which the two objectives are scalarised using a weighted sum $f(x, \lambda) = \lambda f^1(x) + (1 - \lambda) f^2(x)$ where $\lambda \in [0, 1]$, which is straightforward. Thus, for a given weight, a nondominated supported point is obtained. A complete application of a dichomotic method [2] would be possible. However, we have chosen to compute a limited subset of these, for two reasons. Firstly, as a box may be pruned during this refining procedure of the paving, it would be a loss of time to compute a large set only to subsequently delete it. Secondly, the points calculated will be used in the various tests carried out as part of the procedure, and a high number of points will lead to numerous tests. A heuristic choice is made here, in which a number of supported nondominated points are computed using a truncated dichotomic method. This results in a restricted number of computed points within a box.

Looking again at Figure 10, one might be tempted to believe that the boxes closest to the origin are good candidates for reducing or eliminating other boxes. Based on a preliminary numerical experiment, this strategy did not prove particularly successful. The approach adopted therefore is to make a pairwise comparison between all the boxes in the paving, in the order resulting from the branch-and-bound algorithm.

4.2 The `improvePaving` procedure

The `improvePaving` procedure (see Algorithm 3) gets the set of boxes resulting from the initial paving. It starts by considering that all boxes are unpruned (line 3).

First, it computes $S^{\mathcal{B}_0}$ for $\mathcal{B}_0$ and $S^{\mathcal{B}_1}$ for $\mathcal{B}_1$, two sets of supported nondominated points, sorted by increasing order of f_1 (lines 4∼7). Next, all pairs $(\mathcal{B}_0, \mathcal{B}_1)$ of boxes are considered by the `tryToShrinkPruneBox` procedure (lines 8∼10), differentiating between cases where J_1 of $\mathcal{B}_0$ is a singleton and those where it is not. For a given pair $(\mathcal{B}_0, \mathcal{B}_1)$, $\mathcal{B}_0$ is the reference box, and $\mathcal{B}_1$ the candidate box to be pruned or shrank. The points known in $\mathcal{B}_0$ are used to attempt either to shrink the box $\mathcal{B}_1$, or to eliminate $\mathcal{B}_1$ if its ideal point appears to be dominated. The outcome for $\mathcal{B}_1$ is either *pruned* (eliminated entirely), *shrunk* (its bounds are tightened), or *unchanged* (attempts at both shrinking and pruning are unsuccessful).

Finally, all the boxes marked for pruning are definitively deleted from the paving (line 11).

Algorithm 3 `improvePaving`

1: **input:** $\mathcal{P}$, a paving
2: **output:** $\mathcal{P}$, an improved paving
3: $\texttt{pruned}[\mathcal{B}] \leftarrow \texttt{false},\ \forall \mathcal{B} \in \mathcal{P}$
4: **for all** $\mathcal{B} \in \mathcal{P}$ **do**
5: $\quad S^{\mathcal{B}} \leftarrow \texttt{computeSupportedPoints}(\mathcal{B}) \cup \{y^{12}, y^{21}\}$ ▷ some supported points
6: $\quad \texttt{sort}(S^{\mathcal{B}} \updownarrow)$ in ascending order of f_1
7: **end for**
8: **for all** $(\mathcal{B}_0, \mathcal{B}_1) \in \mathcal{P},\ \mathcal{B}_0 \neq \mathcal{B}_1,\ \neg\texttt{pruned}[\mathcal{B}_0],\ \neg\texttt{pruned}[\mathcal{B}_1]$ **do**
9: $\quad \texttt{tryToShrinkPruneBox}(\mathcal{B}_0 \downarrow, \mathcal{B}_1 \updownarrow, \texttt{pruned} \updownarrow)$
10: **end for**
11: $\texttt{deleteBoxes}(\texttt{pruned}\downarrow, \mathcal{P} \updownarrow)$ ▷ delete pruned boxes

4.3 The `tryToShrinkPruneBox` procedure

A box $\mathcal{B}$ has two notable points, $y^{12}(\mathcal{B})$ at the north-west corner and $y^{21}(\mathcal{B})$ at the south-east corner, which define its boundaries (see Definition 2). One functionality of the procedure aims to reduce of a box's bounds without losing any solutions. Its principle is to use two reduction operators that utilize a set of nondominated points corresponding to feasible solutions.

Starting with a pair of boxes $(\mathcal{B}_0, \mathcal{B}_1)$, the procedure `tryToShrinkPruneBox` will apply two reduction operators on $\mathcal{B}_1$, called North-West and South-East, by successively pushing its North-West corner downwards and to the right, then its South-East corner upwards and to the left. Both operators make use of the points $S^{\mathcal{B}_0}$, sorted in ascending order by f_1, to detect an exploitable dominance. At the end of a reduction operator, the procedure checks whether $\mathcal{B}_1$ can be removed entirely. The procedure proceeds in two steps.

Step 1: apply the operator North-West if $y_1^{12}(\mathcal{B}_1) \geq y_1^{12}(\mathcal{B}_0)$:

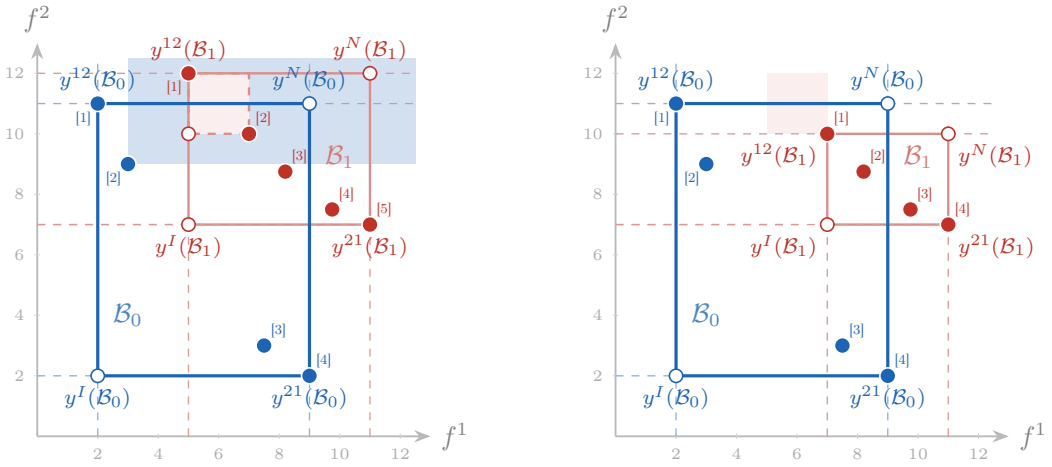
1. Try to shrink $\mathcal{B}_1$ from the North-West, using points of $S^{\mathcal{B}_0}$.
2. Try to prune $\mathcal{B}_1$ using a point of $S^{\mathcal{B}_0}$; if successful, stop.

Explanations:
We aim to prove that a north-western region of $\mathcal{B}_1$ is dominated by known points in $\mathcal{B}_0$. Starting from the point $y^{12}(\mathcal{B}_1)$ and the point $y' = \texttt{next}(S^{\mathcal{B}_1})$, its adjacent point in $S^{\mathcal{B}_1}$, a *local ideal point* y^I_{loc} is obtained. If a point $S^{\mathcal{B}_0}$ dominates this local ideal, the box $\mathcal{B}_1$ can be shrunk:

1. $y^{12}(\mathcal{B}_1) \leftarrow y'$;
2. the points $y^I(\mathcal{B}_1)$ and $y^N(\mathcal{B}_1)$ are updated, thereby contracting the box;
3. the reduction test continues on the shrunk $\mathcal{B}_1$.

The north-west reduction test for $\mathcal{B}_1$ ends as soon as the test is no longer applicable or all candidates (in $\mathcal{B}_0$ or $\mathcal{B}_1$) have been exhausted. Figure 11 illustrates this situation, in which the box $\mathcal{B}_1$ is shrunk. In this example, an inspection of Figure 11b reveals that the box $\mathcal{B}_1$ cannot be further contracted using this operator. Indeed, the new local ideal point y^I_{loc} is not dominated by any point in $S^{\mathcal{B}_0}$.

After attempting to contract the box $\mathcal{B}_1$ via the north-west corner, the procedure checks whether the box can be deleted. The test checks whether the ideal point $y^I(\mathcal{B}_1)$ — which is a lower bound on the best achievable outcome in $\mathcal{B}_1$ — is dominated by a point in $S^{\mathcal{B}_0}$. Figure 12 illustrates this situation.



(a) Before a reduction of $\mathcal{B}_1$

(b) After a reduction of $\mathcal{B}_1$

Figure 11: Boxes $\mathcal{B}_0$ and $\mathcal{B}_1$ with their respective sets of supported nondominated points. $S^{\mathcal{B}_0}$ contains 4 points and $S^{\mathcal{B}_1}$ initially contains 5 points, all sorted in ascending order by f^1. The numbers in square brackets indicate the order in which the points are considered.

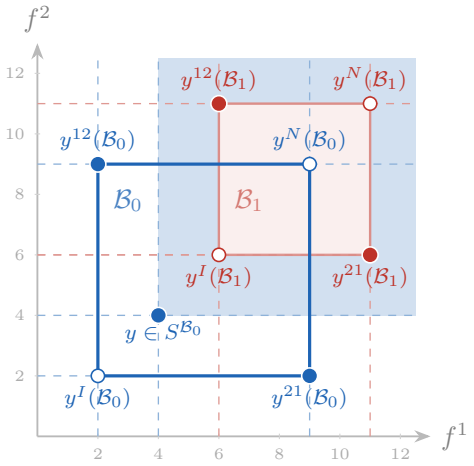


Figure 12: Pruning $\mathcal{B}_1$ with $y \in S^{\mathcal{B}_0}$.

As soon as the test is true, $\mathcal{B}_1$ is marked as pruned and the procedure `tryToShrinkPruneBox` is finished for $\mathcal{B}_1$. Otherwise the step 2 is triggered.

Step 2: apply the operator South-East if $y_2^{21}(\mathcal{B}_1) \geq y_2^{21}(\mathcal{B}_0)$:

1. Try to shrink $\mathcal{B}_1$ from the South-East, using points of $S^{\mathcal{B}_0}$.
2. Try to prune $\mathcal{B}_1$ using a point of $S^{\mathcal{B}_0}$; if successful, stop.

Explanations:
The operations are the symmetry of those described for the north-west, but performed from the south-east corner of the box. In a nutshell, the procedure attempts to move $y^{21}(\mathcal{B}_1)$ to the top-left corner by traversing $S^{\mathcal{B}_0}$ in reverse. The local ideal point y^I_{loc} is built starting from $y^{21}(\mathcal{B}_1)$ and the point $y' = \texttt{prec}(S^{\mathcal{B}_1})$, its adjacent point in $S^{\mathcal{B}_1}$. If a point $S^{\mathcal{B}_0}$ dominates this local ideal, the box $\mathcal{B}_1$ can be shrunk:

1. $y^{21}(\mathcal{B}_1) \leftarrow y'$;

2. the points $y^I(\mathcal{B}_1)$ and $y^N(\mathcal{B}_1)$ are updated, thereby shrinking the box;

3. the reduction test continues on the contracted $\mathcal{B}_1$.

The test for the reduction of $\mathcal{B}_1$ using the South-East operator terminates under conditions that are symmetric to those of the North-West operator.

Case when J_1 of $\mathcal{B}_0$ is a singleton All the remarkable points on a box are joined together into a single point. Consequently, $S^{\mathcal{B}_0}$ is reduced to a single point, for example $y^{12}(\mathcal{B}_0)$. The operators described remain valid but have been simplified: each dominance test is performed exclusively on $y^{12}(\mathcal{B}_0)$, and any failed test results in an immediate termination since there is nothing else to test.

4.4 Didactic example (continued)

The refinement procedure is applied to the paving resulting from the branch-and-bound algorithm. In this case, three boxes containing one facility ({1}, {3}, {5}) and two boxes with two facilities ({1,3}, {1,5}) are present (Figure 9).

Supported nondominated points are then calculated: three for the box {1,3} and two for {1,5} (Figure 13).

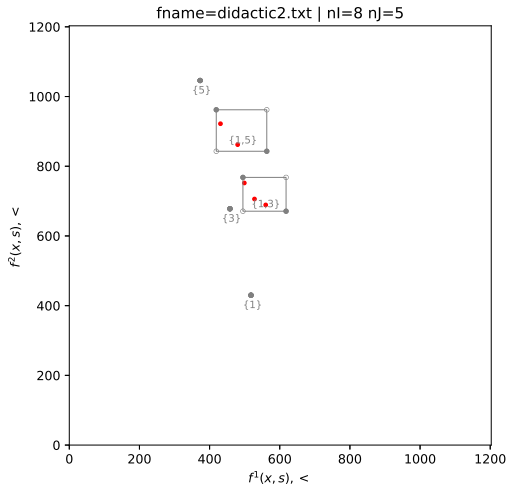


Figure 13: Boxes with supported nondominated points.

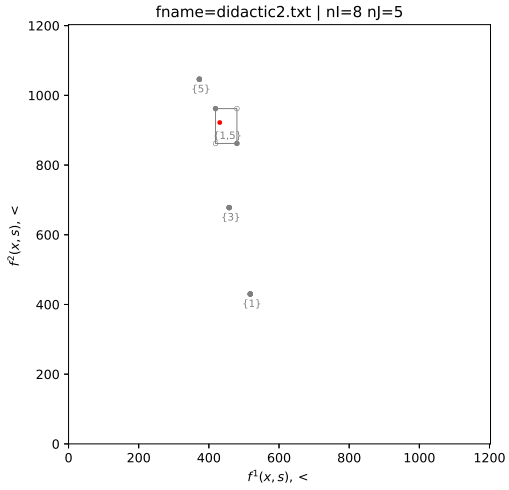


Figure 14: Refined paving $\mathcal{P} = \Big\{\{1\},\{3\},\{5\},\{1,5\}\Big\}$ obtained.

After applying the refinement operators (Figure 14), box $\{1,3\}$ is pruned (the result of applying both operators) and box $\{1,5\}$ is shrunk (the result of applying the south-east operator).

5 Phase 3: the generation

The third phase of the algorithm completes the determination of the set of nondominated points Y_N for the 2-UFLP. This is done by exploring the boxes that compose the improved paving. Each box is associated to a unique set J_1 of open facilities, and to a bi-objective allocation problem. The solution of this problem is presented below, as well as the way the explorations of different boxes are articulated.

5.1 Labeling algorithm

The method applied to solve bi-objective allocation problems is close to the one proposed by Fernández and Puerto [14] and by Stiglmayr et al. [23]. The problem is reformulated as a shortest path problem. Basically, each customer $i \in I$ must be assigned to a facility $j \in J_1$, and the assignment of each customer is independent. The problem to minimize the assignment cost of one customer $i \in I$ can be immediately written as a shortest path problem. Given a graph defined by a source node s_i, $|J_1|$ nodes $s_{i,j}$ and one destination node t_i, $|J_1|$ outgoing arcs from s_i joins the nodes $s_{i,j}$ with a cost of $c_{i,j}$ and represents the possible assignment of the customer i to the facility j. Next, there is one outgoing arc from each node $s_{i,j}$ for $j \in J_1$ to t_i with a $(0,0)$ cost. A shortest path formulation of the bi-objective allocation problem consists simply in concatenating the graphs associated to each customer in any order. This can be done by fusing the nodes t_i and s_{i+1} for all $i \in \{1, \ldots, m-1\}$. An illustration of such a graph is given by Figure 15.

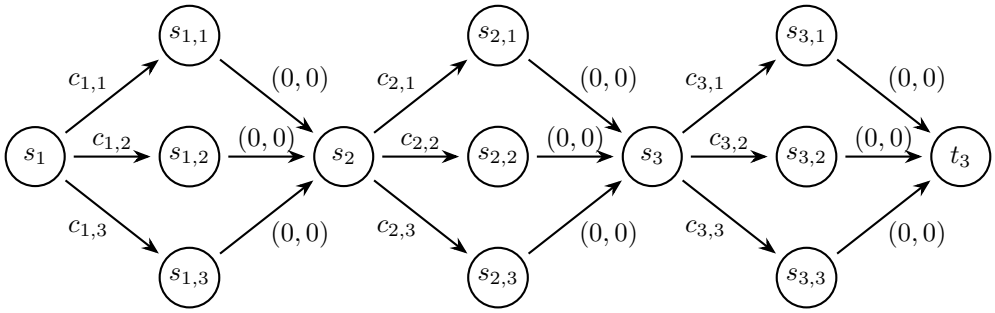


Figure 15: Example of the shortest path formulation of a bi-objective allocation problem with 3 customers and 3 facilities.

Finally, it has been shown by [14] and by [23] that in each subgraph associated to each customer $i \in I$, dominated assignments, i.e. assignments with a cost c_{ij} dominated by another cost $c_{ij'}$ of a same subgraph, cannot be used to define an efficient solution. Hence, the node s_{ij} associated to a dominated assignment is deleted with its entering and outgoing arcs.

There are next some differences with the propositions of [14] and [23]. We exploit the fact that one box $\mathcal{B}$ has been associated to a bi-objective allocation problem, and its size may have been reduced in the refinement algorithm. In other words, we may not solve completely a bi-objective allocation problem, as we may already know that only a subset of its efficient solutions may be of interest for the 2-UFLP problem. The up-right point of the box $\mathcal{B}$ defines upper bounds on each objective for solutions that may be finally efficient for 2-UFLP. These upper bounds can be used to stop the exploration of paths which exceed one of them before reaching the final node t_m.

The main computational cost of a labeling algorithm comes from the filtering dominance of labels. We exploit as much as possible the particular structure of the graph, and also the natural order of nondominated points in the bi-objective case, in order to reduce the number of necessary dominance tests. Given a customer $i \in I$, we suppose that all the labels of nodes s_i are generated, before to generate all the labels of nodes $s_{i,j}$ for $j \in J_1$, and that next, all the labels of nodes s_{i+1} are generated. No dominance test is necessary to generate the labels of nodes $s_{i,j}$ for $j \in J_1$, as these nodes have only one entering arc. Dominance tests are only necessary to obtain the labels of node s_{i+1}, and this corresponds to a dominance filtering between all the labels of nodes $s_{i,j}$ for $j \in J_1$.

The principle we follow is similar to the one initially proposed by Captivo et al. [4] to solve bi-objective knapsack problem: the labels of node s_{i+1} are generated by increasing value of f^1/decreasing value of f^2, i.e. in the lexicographic order. The only difference is that the node s_{i+1} has $|J_1|$ entering arcs, while nodes have at most two entering arcs in the case of knapsack problem. We suppose that the labels of node s_i and therefore the labels of nodes $s_{i,j}$ for $j \in J_1$

are sorted in the lexicographic order. We initially compare the first label of each node $s_{i,j}$ for $j \in J_1$ to get the smallest in the lexicographic sense. This gives immediately the first label of node s_{i+1}. Suppose that we have obtained this label from the node $s_{i,j'}$, we compare next the first label of each node $s_{i,j}$ for $j \in J_1 \setminus \{j'\}$ and the second label of node $s_{i,j'}$ to get again the smallest in the lexicographic sense. This new label is only candidate to be a label of node s_{i+1}, we must first check if it is not weakly dominated by the first label of node s_{i+1}. If it is not (weakly) dominated, it is definitively inserted in the set of labels of node s_{i+1}. This principle is repeated while there are labels to consider in the nodes $s_{i,j}$ for $j \in |J_1|$. Using a binary heap of size $|J_1|$, we obtain the lexicographic minimum between $|J_1|$ labels in a $O(\log_2(|J_1|))$ computational cost. Next, to test if the obtained candidate label is weakly dominated by a label already in s_{i+1} requires only a comparison with the last inserted label, thanks to the order of generation, and only a comparison of the f^2 value is necessary. The overall computational cost for the assignments of customer i is therefore in $O(k|J_1|\log_2(|J_1|))$ where k is the number of labels in node s_i.

5.2 Strategy over the set of boxes

Each time a box is explored, the set of known feasible points U of 2-UFLP filtered by dominance may be updated. Consequently, the new points of U can be used to filter other boxes, if their ideal point is weakly dominated by one of these new points. The order of exploration of the boxes may therefore have an impact on the total computational time.

Heuristics could therefore be proposed for the order of explorations of the boxes. However, in the current design of the algorithm, no particular order is defined for this, the boxes are simply explored in the order in which they are created. According to our experimental resultats, this already gives very good computing times.

5.3 Didactic example (continued)

The labeling algorithm is applied to each surviving box. Here, the allocation subproblem associated to $J_1 = \{1, 5\}$ is considered. After the refinement algorithm, the local nadir point of the associated box is $(480, 962)$.

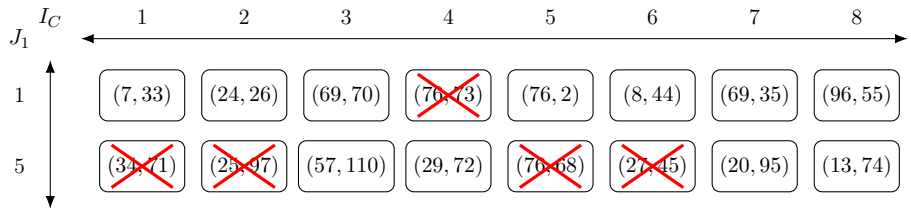
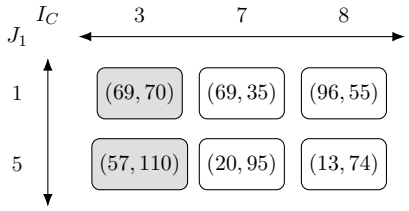


Figure 16: Step 0. Individual assignment costs of all customers to opened facilities corresponding to $J_1 = \{1, 5\}$. The red crosses mark the costs that are dominated.

For customers $1, 2, 4, 5, 6$, there is a dominated assignment, and as there are only two open facilities, the assignment of these customers is straight-forward: customers $1, 2, 5, 6$ are assigned to facility 1, and customer 4 is assigned to facility 5 in all the efficient solutions of this allocation problem.

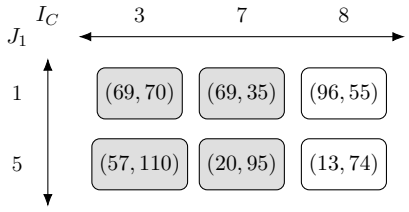
The remaining customers are $I_C = \{3, 7, 8\}$. Before to start the labeling algorithm, the costs of the definitively fixed variables includes the operating cost $(185, 506)$ and the cost $(144, 177)$ of the assignments of customers $1, 2, 4, 5, 6$, for a total of $(329, 683)$. The labeling algorithm considers first customer 3 and generates two labels corresponding to the possible assignments.



$$\{ \quad (386, 793), (398, 753) \quad \}$$

Figure 17: Step 1. Customer 3 has two potential assignments.

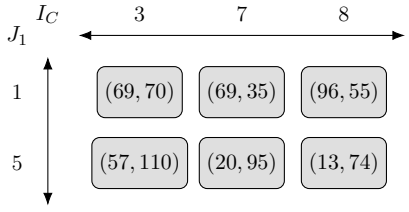
Customer 7 is considered next with its two possible assignments. The labels $(406, 888)$, $(418, 848)$, $(455, 828)$, $(467, 788)$ are generated in the lexicographic order and none of them is weakly dominated.



$$\{ \quad (406, 888), (418, 848), (455, 828), (467, 788) \quad \}$$

Figure 18: Step 2. Customer 7 has two potential assignments.

Finally, customer 8 is considered with its two possible assignments. The labels are generated in the lexicographic order: $(419, 962)$, $(431, 922)$, $(468, 902)$, $(480, 862)$. None of these four first labels are dominated. Next $(502, 943)$ is generated and it is dominated by $(480, 862)$ as $943 > 862$. We can also note that from this last label, the f^1 values of the generated labels have a larger f^1 value than the local nadir of the shrunken box is $(480, 962)$. Consequently, it is not necessary to continue the generation of the next labels.



$$\{ \quad (419, 962), (431, 922), (468, 902), (480, 862) \quad \}$$

Figure 19: Step 3. Customer 8 has two potential assignments.

The obtained labels correspond respectively to the following customer-facility assignments.

$$3 \mapsto 5 \quad 7 \mapsto 5 \quad 8 \mapsto 5$$
$$3 \mapsto 1 \quad 7 \mapsto 5 \quad 8 \mapsto 5$$
$$3 \mapsto 5 \quad 7 \mapsto 1 \quad 8 \mapsto 5$$
$$3 \mapsto 1 \quad 7 \mapsto 1 \quad 8 \mapsto 5$$

This ends the labeling algorithm. Among the generated labels, $(468, 902)$ and $(480, 862)$ are dominated by $(458, 678)$ which is the unique point associated to the set of open facilites $\{3\}$. Finally, two labels remain: $\{(419, 962), (431, 922)\}$. They correspond respectively to the following customer-facility assignments.

$$1 \mapsto 1 \quad 2 \mapsto 1 \quad 3 \mapsto 1 \quad 4 \mapsto 5 \quad 5 \mapsto 1 \quad 6 \mapsto 1 \quad 7 \mapsto 5 \quad 8 \mapsto 5$$

$$1 \mapsto 1 \quad 2 \mapsto 1 \quad 3 \mapsto 5 \quad 4 \mapsto 5 \quad 5 \mapsto 1 \quad 6 \mapsto 1 \quad 7 \mapsto 5 \quad 8 \mapsto 5$$

The final numerical results reported by the proposed algorithm are given below.

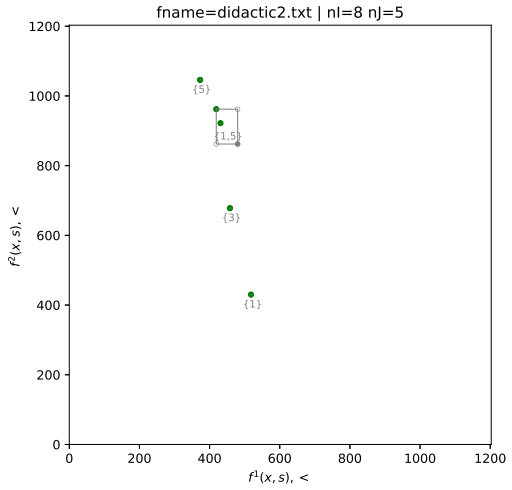


Figure 20: The final paving and Y_N.

costs (f^1, f^2)	facilities opened	allocation user-facility 1, 2, 3, 4, 5, 6, 7, 8
(518, 430)	1	1, 1, 1, 1, 1, 1, 1, 1
(458, 678)	3	3, 3, 3, 3, 3, 3, 3, 3
(373, 1046)	5	5, 5, 5, 5, 5, 5, 5, 5
(419, 962)	1,5	1, 1, 5, 5, 1, 1, 5, 5
(431, 922)	1,5	1, 1, 1, 5, 1, 1, 5, 5

Table 3: Points of Y_N and solutions of X_{E_m}.

Figure 20 shows the final paving and the nondominated points. Table 3 reports the set Y_N of nondominated points and a minimum complete set X_{E_m} of efficient solutions.

6 Numerical experiments

6.1 Datasets and computational environment

Two collections of datasets already used as benchmark in the literature [14, 19] are selected and adapted for evaluating the performance of the proposed algorithm. They are representative of capacitated facility location problems. The additional data provided (allowing to take into account two additional capacity constraints, giving the geographical description of the facilities and the customers, etc.) are ignored in our use.

Dataset F [14]: collection of instances obtained by pairing two single-objective UFLP instances. From this collection, only the 28 instances with the larger dimensions have been selected for our use. Parameters: $n = 30$ customers, $m = 90$ facilities; objective correlation ≈ 0.0; cost ranges $c_{ij}^k \in [0, 100]$, $r_j^k \in [200, 28000]$. These numerical instances are prefixed by "F".

Dataset H [19]: 5 instances representative of real situations in the context of green logistics [18, 19, 22] where cost and CO_2 emissions are the objective functions to minimize simultaneously. Parameters: $n = 10$ customers, $m \in \{2000, 4000, 6000, 8000, 10000\}$ facilities; objective correlation ≈ 0.99; cost ranges $c_{ij}^k \in [0, 43000]$; unique opening costs r_j^k per instance. These numerical instances are prefixed by "H".

Both datasets are available on the vOptLib repository (`https://github.com/vOptSolver/vOptLib`). To the best of our knowledge, these instances have never been used by a algorithm to compute Y_N. Indeed, in their paper, Fernandez and Puerto [14] used instances with a maximum of 20 facilities and 50 customers to evaluate their algorithm. For their part, the H instances were used by Harris et al. in [18, 19] only with metaheuristic-based algorithms, which therefore returned an approximation of Y_N.

The computer used to conduct experiments is a MacBook Pro, equipped with a processor Apple M2 Pro and a memory of 32 Gb, running under macOS 14.6.1. The algorithm is coded in Julia version 1.12.5. The code is available as open source on the GitHub repository at `https://github.com/xgandibleux/biUFLP`. The solver used for solving MILP problems is Gurobi Optimizer version 13.0.

6.2 Results collected on Datasets F and H

The proposed algorithm is evaluated on the two collections of numerical instances. In order to evaluate its effectiveness, it is compared with a generic algorithm and a specific algorithm. The former is an implementation of the ϵ-constraint algorithm (see Listing 2) using Gurobi Optimizer as the MIP solver, and the latter is the algorithm proposed in [14][1]. The numerical experiments were carried out on the same computer, with a maximum computation time limited to 10 minutes per instance.

Results collected are summarized into Table 4 which reports, for each instance, the paving time (t_1), the refinement time (t_2), the labeling time (t_3), the total time taken by the three phases (t_{123}), the number of boxes after the phase 1 ($\#\mathcal{B}_{Pav}$) and after the phase 2 ($\#\mathcal{B}_{Ref}$), the number of nondominated points ($\#Y_N$), the total time taken by the ϵ-constraint algorithm with Gurobi (t_ϵ), and the total time taken by the algorithm of Fernandez-Puerto [14] (t_{FP}). All times reported are given in seconds.

[1]The authors of [14] have kindly accepted to provide us the original source code in Vax Fortran, implementing the algorithm described in their paper. Compiling and running were success, thus the unaltered code has been executed in our numerical environment. However, the size of the datasets used in our experiments is larger than the datasets used by the authors in 2003. As it, this code cannot manage the collection 'H' (due to the size of the instances) and encounters numerical issues for several instances from collection 'F' (reported as 'N.A.' in Table 4).

Instances	Three-Phase algorithm							ϵ-const.	[14]
	t_1	t_2	$\#\mathcal{B}_{Pav}$	$\#\mathcal{B}_{Ref}$	t_3	$\#Y_N$	t_{123}	t_ϵ	t_{FP}
F50-51	0.1655	0.0878	152	10	0.0357	1229	0.2890	270.1916	N.A.
F50-52	0.0075	0.0226	11	5	0.0116	408	0.0417	103.6391	44.9917
F50-53	0.1124	0.0185	21	7	0.0099	771	0.1408	162.9394	47.8149
F50-54	0.0078	0.0160	16	13	0.0120	700	0.0358	175.1433	N.A.
F50-55	0.0075	0.0112	13	9	0.0393	513	0.0580	110.5132	62.7686
F50-56	0.0082	0.0206	23	17	0.1381	729	0.1669	331.3211	N.A.
F50-57	0.0070	0.0173	17	14	0.0107	616	0.0350	240.9657	N.A.
F51-52	0.0107	0.1218	38	15	0.0322	635	0.1647	159.1519	N.A.
F51-53	0.0118	0.1229	32	9	0.0429	1047	0.1776	249.3033	50.7325
F51-54	0.0108	0.0219	24	19	0.1500	1013	0.1827	273.2615	N.A.
F51-55	0.0147	0.0253	31	22	0.1324	1111	0.1724	427.6619	N.A.
F51-56	0.0076	0.0153	14	12	0.0104	755	0.0333	159.0428	N.A.
F51-57	0.0076	0.1081	18	12	0.0363	796	0.1520	227.1591	N.A.
F52-53	0.0017	0.0047	14	8	0.0203	435	0.0267	103.5365	23.7430
F52-54	0.0006	0.0009	8	8	0.1208	47	0.1223	3.3449	45.4852
F52-55	0.0006	0.0005	3	3	0.0458	20	0.0469	2.9087	19.8469
F52-56	0.0006	0.0005	6	6	0.0091	15	0.0102	1.8619	36.7843
F52-57	0.0006	0.0009	6	6	0.1174	16	0.1189	1.5085	N.A.
F53-54	0.0009	0.0011	7	7	0.0118	333	0.0138	24.5483	32.7435
F53-55	0.0013	0.0010	6	6	0.0273	306	0.0296	69.1338	24.3891
F53-56	0.0009	0.0010	5	4	0.0118	318	0.0137	37.5853	37.7964
F53-57	0.0011	0.0005	9	9	0.1203	173	0.1219	31.3258	45.3409
F54-55	0.0003	0.0003	5	5	0.0145	37	0.0151	4.1285	N.A.
F54-56	0.0002	0.0004	4	4	0.0414	22	0.0420	3.7008	18.7721
F54-57	0.0002	0.0002	8	8	0.0097	20	0.0101	2.2026	39.9407
F55-56	0.0003	0.0000	5	5	0.1420	5	0.1423	0.2877	23.3804
F55-57	0.0002	0.0000	4	4	0.0080	4	0.0082	0.1792	19.3553
F56-57	0.0001	0.0000	3	3	0.0114	3	0.0115	0.1564	26.1427
H10-2000	0.0406	0.1922	11	11	0.0755	13412	0.3083	T.O.	N.A.
H10-4000	0.0794	0.4665	11	11	0.4475	69167	0.9934	T.O.	N.A.
H10-6000	0.1064	0.7579	13	13	1.1836	172278	2.0479	T.O.	N.A.
H10-8000	0.1495	1.3330	12	12	16.0485	731385	17.5310	T.O.	N.A.
H10-10000	0.1831	1.7360	12	12	7.9293	4931	9.8484	T.O.	N.A.

Table 4: Computational results. All times reported are given in seconds. T.O.: time out after 600 seconds. N.A.: not available. t_1: paving time. t_2: refinement time. $\#\mathcal{B}_{Pav}$: number of boxes after phase 1. $\#\mathcal{B}_{Ref}$: number of boxes after phase 2. t_3: labeling time. t_{123}: total time taken by the three phases algorithm. $\#Y_N$: number of nondominated points. t_ϵ: total time taken by the ϵ-constraint algorithm with Gurobi. t_{FP}: total time taken by the Fernandez-Puerto algorithm.

6.3 Analysis of numerical results

The three-phase algorithm, the ϵ-constraint algorithm using Gurobi, and the Fernandez-Puerto algorithm were experimented on all instances F and H.

A comparison of the performance of the three algorithms. Firstly, only the three-phase algorithm was able to process all instances within the limited computation time; in particular, it is the only one capable of handling the H instances. Furthermore, the three-phase algorithm proved to be the most efficient across all instances, with a difference in efficiency that can be huge. For example, on the instance F51-53, the Fernandez-Puerto algorithm and the ϵ-constraint algorithm are respectively 285 and 1403 times slower. In terms of computational time, the advantage clearly goes to the 3-phase algorithm.

There are two reasons for this. With regard to the ϵ-constraint algorithm, given the very high number of nondominated points generally found in these instances (see column Y_N in Table 4), and given that for each point of Y_N there is at least one sub-problem to be solved using a MIP solver, even with a highly efficient solver this inevitably leads to prohibitively large solution time. With regard to the Fernandez-Puerto algorithm, it is difficult to determine whether the observed performance is due to the solution algorithm itself or to the implementation choices. In particular, we therefore do not exclude the possibility that, after a retrofactoring of their implementation —which is outside the scope of this paper— more favorable solution times might be observed, as well as better numerical stability when faced with large instances.

An in-depth look at the algorithm in three stages. Secondly, each of the three phases – paving, refinement and generation – requires extremely short computation times. Unsurprisingly, for instances with a large number of nondominated points —which is particularly the case for H-instances— the generation phase is the most time-consuming due to the amount of work it involves in such cases. Although this concerns the solution of allocation problems —a problem that has been widely studied in the literature—, the care taken (1) in our algorithm, which exploits all the properties of this problem, and (2) in its coding, with particular attention paid to performance engineering, means that our Phase 3 proves to be exceptionally effective when there is a large number of nondominated points to be calculated.

Numerical experiments show that the number of boxes obtained at the end of phase 1 is low, particularly when compared to the $2^n - 1$ possible boxes, and this number is often reduced after the application of phase 2. This is observed for the two sets of instances. However, phases 1 and 2 may prove unsuccessful on very specific instances. Apart from pathological instances, these phases do indeed filter the decision space by providing a relevant paving. Heuristic choices have been made to achieve a balance between the algorithm's computational cost and its performance; consequently, not all possible information and strategies are used during these two phases.

Thirdly, and in terms of decision support, the obtained paving provides relevant information that can be presented to a decision-maker in order to collect their preferences and guide the resolution process within the framework of an interactive method. This is fully conceivable given the very short computational times of the three phases, and this approach is without loss of information, since there are no nondominated point outside of the paving. For a decision maker facing to real situations (such as green logistics), the boxes provide an useful information for its learning about nondominated points without generating until several thousands of nondominated points. A human-computer dialog for such an optimization problem may be designed around the following protocol: in order to select a point, the decision-maker may navigate [1] between boxes, compare and analyze points generated for some selected boxes, giving him a close view only on relevant options according his preferences.

A look at the plot of nondominated points in the objective space. The number of nondominated points can be extremely high within a box, as shown in Figures 21 and 22, for instances F and H respectively. This is particularly true for instances H, where up to 493 058 nondominated points are distributed across only 11 boxes. Furthermore, the boxes may or may not overlap with one another in the objective space. For example, for F50-51, the boxes generally overlap, meaning that the nondominated points appear to be distributed more or less uniformly (see Figure 21), whereas for F50-56 (Figure 23) or for H10-2000 (Figure 22) the nondominated points appear in distinct clusters in the objective space. Finally, the linear relaxation of the 2-UFLP can be very good, as in case 50-51 (not shown), or very poor, as in case F50-56 (Figure 23). In this last case, an enumerative algorithm using the linear relaxation as a bound set would be ineffective.

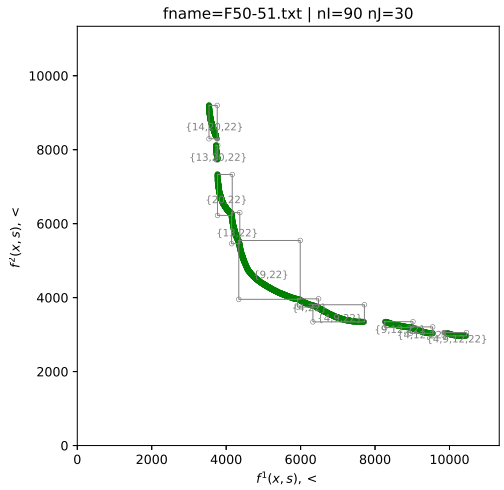


Figure 21: Final paving obtained and Y_N for F50-51.

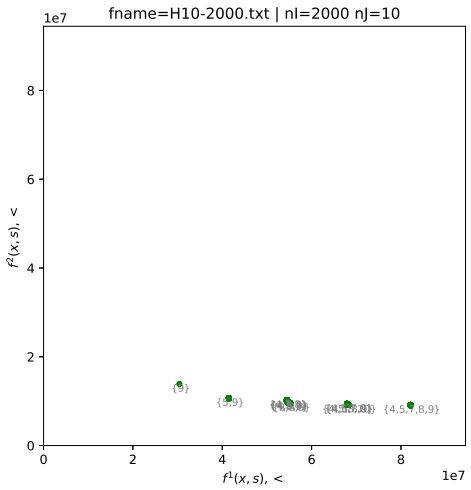


Figure 22: Final paving obtained and Y_N for H10-2000.

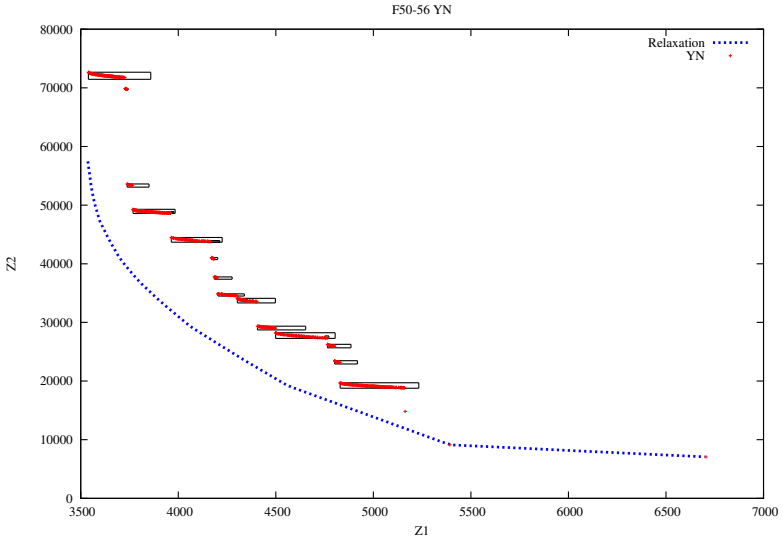


Figure 23: Final paving obtained, Y_N and lower bound set for the instance F50-56. The dotted blue line shows the linear relaxation.

7 Acknowledgments

We would like to express our special thanks to Professor E. Fernández (Universitat Politècnica de Catalunya, Spain), who was kind to provide us the Fortran source code of the algorithm published in 2003 and the data sets F. Also, We would like to thank Professor Christine L. Mumford (Cardiff University, UK) for having provided the data sets H for our numerical experiments. Finally we would also like to thank Salim Bourougaa, Alban Derrien and Axel Grimault, who, whilst they were Master's students, implemented in C++ the initial version of this algorithm.

References

[1] Richard Allmendinger, Matthias Ehrgott, Xavier Gandibleux, Martin Josef Geiger, Kathrin Klamroth, and Mariano Luque. Navigation in multiobjective optimization methods. *Journal of Multi-Criteria Decision Analysis*, 24(1-2):57–70, 2017. doi: https://doi.org/10.1002/mcda.1599. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/mcda.1599`.

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

[3] Fritz Bökler. *Output-sensitive complexity of multiobjective combinatorial optimization with an application to the multiobjective shortest path problem.* PhD thesis, Dortmund University, Germany, 2018. URL `http://hdl.handle.net/2003/37134`.

[4] Maria E. V. Captivo, João C. N. Clímaco, José R. Figueira, Ernesto Martins, and José Luís Santos. Solving bicriteria 0-1 knapsack problems using a labeling algorithm. *Computers & Operations Research*, 30(12):1865–1886, 2003. doi: 10.1016/S0305-0548(02)00112-0. URL `https://doi.org/10.1016/S0305-0548(02)00112-0`.

[5] Gérard Cornuejols, George L. Nemhauser, and Laurence A. Wolsey. The uncapacitated facility location problem. In Pitu B. Mirchandani and R.L. Francis, editors, *Discrete Location Theory*, Discrete Mathematical Optimisation, chapter 9, pages 55–117. Wiley-Interscience, New York, 1990.

[6] Oscar Dowson, Xavier Gandibleux, and Gökhan Kof. `MultiObjectiveAlgorithms.jl`: a julia package for solving multi-objective optimization problems. *INFORMS Journal on Computing (to appear, accepted 04/04/2026)*, 2026. URL `https://arxiv.org/abs/2507.05501`.

[7] Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: A Modeling Language for Mathematical Optimization. *SIAM Review*, 59(2):295–320, 2017. doi: 10.1137/15M1020575. URL `https://doi.org/10.1137/15M1020575`.

[8] Matthias Ehrgott and Xavier Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spectrum*, 22:425–460, 2000. ISSN 0171-6468.

[9] Matthias Ehrgott and Xavier Gandibleux. Bounds and bound sets for biobjective combinatorial optimization problems. In *Multiple Criteria Decision Making in the New Millennium*, volume 507 of *Lecture Notes in Economics and Mathematical Systems*, pages 241–253. Springer Berlin Heidelberg, 2001.

[10] Matthias Ehrgott and Xavier Gandibleux. Multiobjective combinatorial optimization. In Matthias Ehrgott and Xavier Gandibleux, editors, *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Survey*, volume Volume 52 of *International Series in Operations Research and Management Science*, chapter 8, pages 369–444. Kluwer Academic Publishers, 2002.

[11] Matthias Ehrgott and Xavier Gandibleux. Bound sets for biobjective combinatorial optimization problems. *Computers & Operations Research*, 34:2674–2694, 2007. doi: 10.1016/J.COR.2005.10.003. URL `https://doi.org/10.1016/j.cor.2005.10.003`.

[12] Horst A. Eiselt and Gilbert Laporte. Objectives in location problems. In Zvi Drezner, editor, *Facility location: a survey of applications and methods*, Springer series in operations research, chapter 8, pages 151–180. Springer, 1995. ISBN 0387945458.

[13] Reza Zanjirani Farahani, Mehdi SteadieSeifi, and Nasrin Asgari. Multiple criteria facility location problems: A survey. *Applied Mathematical Modelling*, 34:1689–1709, 2010.

[14] Elena Fernández and Justo Puerto. Multiobjective solution of the uncapacitated plant location problem. *European Journal of Operational Research*, 145(3):509–529, 2003.

[15] Xavier Gandibleux. Native multi-objective mathematical programming software, 2026. Book chapter, to appear.

[16] Arthur M. Geoffrion. Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, 22:618–630, 1968.

[17] Eric Gourdin, Martine Labbé, and Hande Yaman. Telecommunication and location - a survey. In Zvi Drezner and Horst W. Hamacher, editors, *Facility Location*, chapter 9, pages 275–305. Springer, 2002.

[18] Irina Harris, Christine L. Mumford, and Mohamed M. Naim. The multi-objective uncapacitated facility location problem for green logistics. In *IEEE Congress on Evolutionary Computation*, pages 2732–2739. IEEE, 2009.

[19] Irina Harris, Christine L. Mumford, and Mohamed M. Naim. An Evolutionary Bi-Objective Approach to the Capacitated Facility Location Problem with Cost and CO_2 Emissions. In *GECCO'11 proceedings, Dublin (Ireland)*, page 8 pages, 2011.

[20] Jakob Krarup and Peter Mark Pruzan. The simple plant location problem: Survey and synthesis. *European Journal of Operational Research*, 12(1):36–81, 1983.

[21] Stefan Nickel, Justo Puerto, and Antonio Rodriguez-Chia. MCDM location problems. In J. Figueira, S. Greco, and M. Ehrgott, editors, *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 78 of *International Series in Operations Research & Management Science*, pages 761–787. Springer, 2005.

[22] Abdelkader Sbihi and Richard W. Eglese. Combinatorial optimization and green logistics. *Annals of Operations Research*, 175:159–175, 2010.

[23] Michael Stiglmayr, José Rui Figueira, and Kathrin Klamroth. On the multicriteria allocation problem. *Annals of Operations Research*, 222:535–549, 2014.