Bi-objective multi-assignment capacitated location-allocation problem

Fouad MAACH

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
In
Industrial and Systems Engineering

Prof. Joel Nachlas

Prof. Subhash C. Sarin

Prof. Bruno Castanier

under the guidance of Dr. Olivier Péton

January 30, 2007
Blacksburg, Virginia

Keywords: Bi-objective combinatorial optimization, load balance, multiobjective optimization evolutionary algorithm

Bi-objective multi-assignment capacitated location-allocation problem

Fouad MAACH

(ABSTRACT)

Optimization problems of location-assignment correspond to a wide range of real situations, such as factory network design. However most of the previous works seek in most cases at minimizing a cost function. Traffic incidents routinely impact the performance and the safety of the supply. These incidents can not be totally avoided and must be regarded. A way to consider these incidents is to design a network on which multiple assignments are performed.

Precisely, the problem we focus on deals with power supplying that has become a more and more complex and crucial question. Many international companies have customers who are located all around the world; usually one customer per country. At the other side of the scale, power extraction or production is done in several sites that are spread on several continents and seas. A strong willing of becoming less energetically-dependent has lead many governments to increase the diversity of supply locations. For each kind of energy, many countries expect to deal ideally with 2 or 3 location sites. As a decrease in power supply can have serious consequences for the economic performance of a whole country, companies prefer to balance equally the production rate among all sites as the reliability of all the sites is considered to be very similar. Sharing equally the demand between the 2 or 3 sites assigned to a given area is the most common way. Despite the cost of the network has an importance, it is also crucial to balance the loading between the sites to guarantee that no site would take more importance than the others for a given area. In case an accident happens in a site or in case technical problems do not permit to satisfy the demand assigned to the site, the overall power supply of this site is still likely to be ensured by the one or two available remaining site(s). It is common to assign a cost per open power plant and another cost that depends on the distance between the factory or power extraction point and the customer. On the whole, such companies who are concerned in the quality service of power supply have to find a good trade-off between this factor and their overall functioning cost. This situation exists also for companies who supplies power at the national scale. The expected number of areas as well that of potential sites, can reach 100. However the targeted size of problem to be solved is 50.

This thesis focuses on devising an efficient methodology to provide all the solutions of this bi-objective problem. This proposal is an investigation of close problems to delimit the most relevant approaches to this untypical problem. All this work permits us to present one exact method and an evolutionary algorithm that might provide a good answer to this problem.

Acknowledgments

Numerous people have helped me throughout the course of this research. I am in debt to them and want to take this opportunity to express my sincerest acknowledgments.

The completion of this dissertation would not have been possible without the guidance and support of my advisor, Dr. Xavier Gandibleux.

I thank Dr. Joel Nachlas and Dr. Subhash C. Sarin for their great involvement and their help during my thesis.

I thank Dr. Olivier Péton for all his advice and attention that were very helpful.

I thank Anthony Przybylski who has spent numerous hours giving me help in computing aspects of this research.

Contents

List of Figures

List of Tables

Context

The multi-assignment problem with minimization simultaneously of the cost and the balance rate presents a wide range of practical applications that were revealed through various lectures:

- **Project management**
  Many prestigious consulting firms employ skilled people in several countries. It is well-known that the cost of such a worker depends on his level of expertise and the number of projects that is assigned to him. For performance reasons, consulting firms prefer to assign every project to several experts as their reputation could not bear a failure in any project. To ensure that a project can be done on time, it is crucial to balance the work charge among the experts. Due to the diversity of country regulations and the level of implication of every expert in a company, the amount of available time can differ greatly between them. Even if many companies still tries to reduce their costs as much as possible, some of them are aware of the importance of the quality of their service and prefers to concentrate on a trade-off between the total cost and the work balance rate.

- **Large volume distribution systems**
  Companies like Wall-Mart (US) or Système U (France) have to design large volume distribution systems which consist usually of warehouses and local shops. Designing an efficient system that provides good performances through all the year is a difficult task. The number of new articles per year and the difficulty to forecast the success level of these articles and to decide in advance the assignment between all the articles and the warehouses lead to an inextricable problem. As it is not economically unrealistic for a warehouse to run at a high level during several months before stopping almost it shipping activities during the rest of the year, it is relevant to focus simultaneously on the total functioning cost and the balance loading between the warehouses. The total average demand for each shop is considered and has to be satisfied equally by two or three warehouses. Such a loading balance prevents from opening two big warehouses whose total capacity would be useful only during a relative small portion of the year. Once the distribution system is fixed, the supply chain manager can make it run smoothly through all the year by storing each item at 2 or 3 warehouses.

Chapter 1

Introduction

1.1 Definition of the problem

The formulation of the bi-objective multi-assignment capacitated location-allocation problem can be expressed as follows:

1. - There are $|J|$ potential sites that can be opened to satisfy the demand from $|I|$ areas.
     - Each site $j \in J$ has a capacity Q_j and a fixed set up cost f_j.
     - The linking cost from an area $i \in I$ to a site $j \in J$ equals to c_{ij}. The total network load from an area i is denoted by d_i.
   - The involved constraints are the following ones:

     Capacity The constraint capacity Q_j has obviously to be respected.

     Balance Each site i has to be connected to N open sites. As a consequence, the traffic load from a open site connectet to the area i equals to $w_i = \frac{d_i}{N}$. Balancing the charge of the sites means minimizing the maximum difference of the percentage of used capacity between each pair of open sites, called *rate*.

   Such a problem is called a *multiple objective problem* as more than one objective function is formulated : in this case, the objectives are the total cost and the rate.

2. The final formulation of this problem will have to take into account:

   - the potential location of the sites and areas,
   - the number N depends on the decision maker and the characteristics of the items that are supplied. However the target values of N can be reduced to {1,2,3}.

Such a problem gathers two types of problem: a problem of location of the new open sites and a problem of assignment between the open sites and the areas. As a consequence, it is necessary to sum up all the potentially relevant exact and heuristic approaches to these kinds of problems. The goal of this thesis is to provide procedures to solve instances that can count until 50 sites and 50 areas. Parallely, efforts will be made to provide a good heuristic method to solve this kind of problems in such a way to approximate all the solutions of our problem. The section 1.4 is the first chapter introduces all the necessary concepts to measure the difficulty of this optimization

problem.

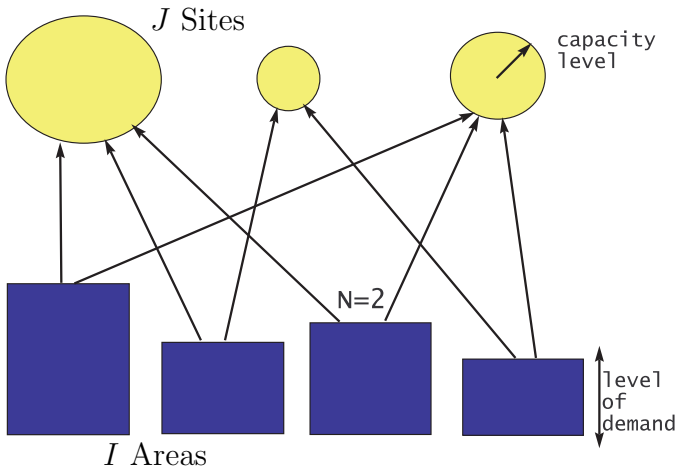


Figure 1.1: Illustration of our problem with $I = 4, J = 3, N = 2$

1.2 Formulation *MMX*

The formulation of this problem *MMX* can be expressed as follows:

$$Min \begin{pmatrix} Cost \\ Rate \end{pmatrix} \tag{1}$$

$$where \quad Cost = \sum_{j=1}^{J} f_j y_j + \sum_{i=1}^{I} \sum_{j=1}^{J} c_{ij} x_{ij}$$
$$Rate = \rho$$

$$\sum_{j=1}^{J} x_{ij} = N \qquad \forall i \in \{1...I\} \tag{2}$$

$$Q_j y_j = \sum_{i=1}^{I} \frac{d_i}{N} x_{ij} + \alpha_j \qquad \forall j \in \{1...J\} \tag{3}$$

$$(-\rho - 2 + y_j + y_{j'}) \leq \frac{\alpha_j}{Q_j} - \frac{\alpha_{j'}}{Q_{j'}} \qquad \forall (j, j') \in \{1...J-1\} \times \{j+1...J\} \tag{4}$$

$$(-\rho - 2 + y_j + y_{j'}) \leq -\frac{\alpha_j}{Q_j} + \frac{\alpha_{j'}}{Q_{j'}} \qquad \forall (j, j') \in \{1...J-1\} \times \{j+1...J\} \tag{5}$$

$$y_j \in \{0,1\}, x_{ij} \in \{0,1\}, \alpha_j \geq 0, \rho \geq 0 \tag{6}$$

The decision variables are respectively y_j and x_{ij}:

- $y_j = \begin{cases} 0 & \text{if the site } j \text{ is closed} \\ 1 & \text{if the site } j \text{ is open} \end{cases}$
- $x_{ij} = \begin{cases} 0 & \text{if the site } j \text{ is assigned to the area } i \\ 1 & \text{if the site } j \text{ is assigned to the area } i \end{cases}$

The constraint (2) ensures that each area is connected to N sites.The constraint (3) expresses the capacity limit for each site. Besides, such an equation ensures that if a site is closed, no area is connected to this site.The constraint (4) and (5) are used to evaluate the imbalance rate. Several formulation for this constraint can be used: $\frac{\alpha_j}{Q_j}$ and $\frac{\alpha_{j'}}{Q_{j'}}$ lie between 0 and 1. As a consequence,

- If the two sites j and j' are closed, *rate* is not constrained.
- If one of the two sites j or j' is closed and the other open, *rate* is superior to the percentage of remaining capacity minus one (or the opposite), i.e. greater than a non positive value.
- If the two sites j and j' are open, the left-hand side member of the inequality equals to *rate* and the other member equals to the difference of the percentage of used capacity between the two sites or the opposite.

The constraints (2) to (6) define a search space denoted X.

1.3 Other formulations of the rate constraint

The constraints (4) and (5) lead to a hard problem to solve in comparison with the problem of minimizing cost. That is why other formulations of this set of constraints have been explored.

A first alternative can be to minimize the range of the proportion of used capacity. Two additional variables are necessary,λ_{min} and λ_{max}, one to provide the smallest proportion λ_{min} and the other one λ_{max} the biggest proportion. The new objective function consists of minimizing the difference between λ_{max} and λ_{min}.

$$\frac{\sum_{i \in I} \frac{d_i}{N} \times x_{ij}}{Q_j} \leq \lambda_{max} \quad \forall j \in \{1, ..., J\} \quad (7)$$

$$\lambda_{min} \leq \frac{\sum_{i \in I} \frac{d_i}{N} \times x_{ij}}{Q_j} + 1 - y_j \quad \forall j \in \{1, ..., J\} \quad (8)$$

And the constraint (3) is slightly changed as the variables α_j are not used any more in this formulation.

$$\sum_{i=1}^{I} \frac{d_i}{N} x_{ij} \leq Q_j y_j \quad \forall j \in \{1, ..., J\} \quad (9)$$

1.4 Fundamental concepts of multi-objective optimization problem

For all the following definitions [Ehr05a], we consider the following bi-objective problem:

$min_{x \in X}(f_1(x), f_2(x))$

1.4.1 Set of efficient solutions and non-dominated points

(see fig. 1.2)

Definition 1. *A feasible point $\widehat{x} \in X$ is called an efficient solution if there is no $x \in X$ such that $f(x) \leq f(\widehat{x})$, i.e. there is no $x \in X$ such that: $f_k(x) \leqq f_k(\widehat{x})$ for $k \in \{1,2\}$ and $f_i(x) < f_i(\widehat{x})$ for $i \in \{1,2\}$.*
If $\widehat{x}$ is efficient, $f(\widehat{x})$ is called non-dominated point.

Definition 2. *The set of all efficient solutions $x \in X$ is called the efficient set and denoted X_E.*

Definition 3. *The set of all non-dominated points is $y = f(x) \in Y$, with $x \in X_E$, is called the non-dominated set and denoted Y_N.*

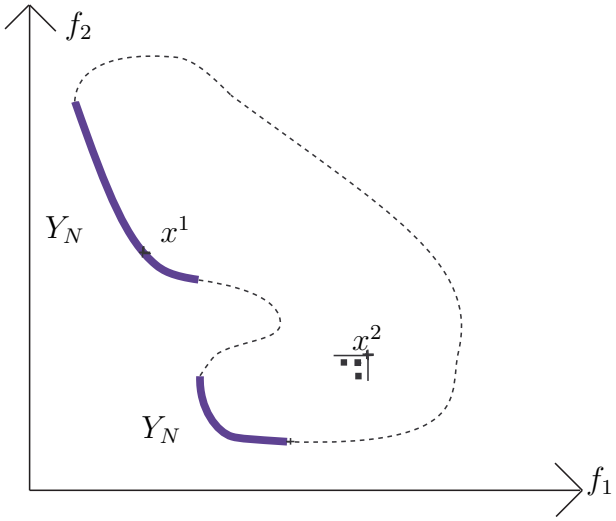


Figure 1.2: Set of non-dominated points: $x^1 \in X_E, x^2 \in X$

1.4.2 Supported/Non-supported efficient solutions

Definition 4. *Supported (efficient) solutions are optimal solutions of a weighted sum single objective problem. All the remaining solutions $x \in X_E$ are non-supported (efficient) solutions.*

$$\min \{\lambda^1 f_1(x) + \lambda^2 f_2(x) : x \in X, \lambda^1 > 0, \lambda^2 > 0\}.$$

All supported nondominated points are located on the "lower-left boundary" of the convex hull of X.

1.4.3 Ideal and Nadir points

(see fig. 1.3)

Definition 5. *The point $Y^I = (Y_1^I, Y_2^I)$ given by $Y_k^I = min_{\{x \in X\}} f_k(x)$, $k \in \{1,2\}$ is called the Ideal point.*

Definition 6. *The point $Y^N = (Y_1^N, Y_2^N)$ given by $Y_k^N = max_{\{x \in X_E\}} f_k(x)$, $k \in \{1,2\}$ is called the Nadir point.*

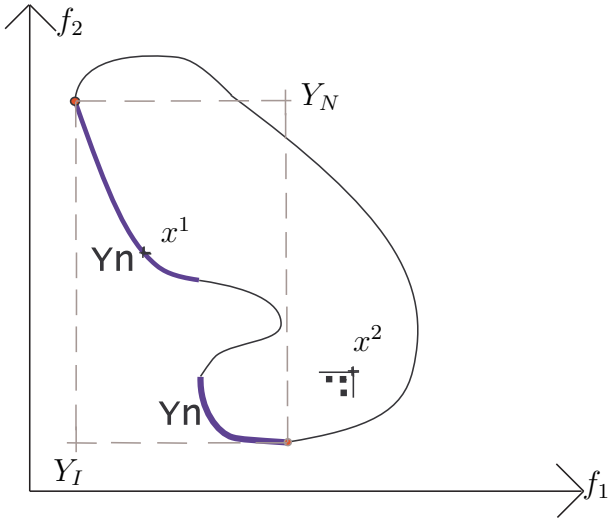


Figure 1.3: Illustration of the Ideal and Nadir points: Y_I, Y_N

1.4.4 Upper and lower bound sets

A lower as well as an upper bound for a mono-objective problem is defined intuitively. In a multiple objective context, it is necessary to introduce the definition of the lower and upper bound sets of the non-dominated points.

- **Notations**
  We shall use $\mathbb{R}^p_{\leqq} = \{y \in R^p \ : \ y \ \geqq 0\}$ and analogously for $\mathbb{R}^p_{\geqq}$ and $\mathbb{R}^p$.
  Let S be a set in R^p:
  S is said to be $\mathbb{R}^p_{\leqq}$-closed if $S + \mathbb{R}^p_{\leqq} = \{s + y \ : \ s \in S, y \in \mathbb{R}^p_{\leqq}\}$ is closed.
  S is said to be $\mathbb{R}^p_{\leqq}$-bounded if $\exists\, s^0 \in R^p, S \subset \{s^0 + R^p\}$.
  The set of non-dominated points of S is denoted S_N and cl(S) denotes the closure of S.

- **Definitions**
  Let us consider the set of non-dominated points Y_N of a multi-objective problem, and a subset $\hat{Y} \subset Y_N$.

  Definition 7. *A lower bound set L for $\hat{Y}$ is an $\mathbb{R}^p_{\leqq}$-closed and $\mathbb{R}^p_{\leqq}$-bounded set $L \subset R^p$ such that $\hat{Y} \subset L + \mathbb{R}^p_{\leqq}$ and $L \subset (L + \mathbb{R}^p_{\leqq})_N$.*

  Definition 8. *A upper bound set U for $\hat{Y}$ is an $\mathbb{R}^p_{\leqq}$-closed and $\mathbb{R}^p_{\leqq}$-bounded set $U \subset R^p$ such that $\hat{Y} \subset cl[(U + \mathbb{R}^p_{\leqq})^c]$ and $U \subset (U + \mathbb{R}^p_{\leqq})_N$.*

  The first inclusion of each of both definitions are intuitive. Nevertheless the only first condition of the lower bound set would allow for $y \in \widehat{Y}$ to have $y < l \, \forall l \in L$. It can be observed that $\{Y^I\}$ is a lower bound set for all $\hat{Y} \subset Y_N$ and $\{Y^N\}$ is an upper bound set for all $\hat{Y} \subset Y_N$.

Chapter 2

Single-objective Facility Location Problem investigation

The facility location problem is the class of problem from which our problem is derived. Hence the results for FLP can be derived to the problem we focus on. A huge wide of papers dedicated to this class of problem is available, as well as two often quoted books,[NP05] and [HD03], and software systems (among them, the LOLA platform include a wide-range of LFP algorithms [DoMUoK06]). The variety of real situations explains why it is expected to find such a rich literature. Without pretending to be exhaustive, this section gathers contributions that are close to our problem and whose solving techniques show a potential interest for our study.

2.1 Exact methods

Erlenkotter [Erl78] proposed the DUALOC procedure to find optimal solutions to the uncapacitated location problem. This method solves the dual problem by using the ascent sub-gradient to improve the solution. More recently, the use of the Lagrangean relaxation method was revisited by Galvao [Gal93a] and Daskin [Das95]. The methods based on a Branch & Bound consist of the main approach to solve exactly a problem. Even if numerous promising results have been obtained so far, the use of these methods on real large-sized problems has revealed their limits on a wide range of problems. That is why approximate methods have been investigated.

2.2 Approximate methods

The approximate methods can be sorted in two main classes:

- those that progress blindly, called greedy algorithm.
- those that progress by exploiting the collected information through the search process, called heuristic and meta-heuristic.

2.3 Greedy algorithm

A relevant approach to generate a good feasible solution are greedy algorithms that allow to generate often a better solution when coupled with an exploration algorithm.

After defining a criterion to sort each candidate, a greedy algorithm can start from an empty solution (whose decision variables equal to 0) to add the best candidate until one constraint is not satisfied, called ADD operation [KH63]. Conversely, the greedy algorithm can start with a complete solution (whose decision variables equal to 1) and removes iteratively the worst candidate until all the constraints are satisfied, called DROP operation [CFN77]. Advanced greedy algorithms combines both DROP and ADD operations.

2.4 Heuristics and Meta-Heuristics

2.4.1 Heuristics

Once a feasible solution is found, there exist several approaches to improve it. About the P-median problem, an efficient and fast approach is that of Maranzana [Mar64] which is based on the ease of solving the **1-median problem**. His algorithm partitions the customers' space by facility, then solves each 1-median problem. As long as a facility changes, the algorithm reiterates the process with a new partitioning.

The Uncapacitated Facility Location problem [UFLP] can be restricted to solving a **set of P-median problems**. In that perspective, Teitz and Bart [TP68] developed an exchange swap algorithm for this kind of problem.

2.4.2 Tabu search [TS]

More sophisticated algorithms have been developed to overcome the limitation of starting with a sub-optimal number of facilities: Tabu search, generally attributed to F. Glover [Glo77], is an important example. It is a way to enhance the performance of a local search method by using memory structures. Tabu search uses a local or neighbourhood search procedure to iteratively move from a solution x to a solution x' in the neighbourhood of x, until some stopping criterion has been satisfied. To explore regions of the search space that would be left unexplored by the local search procedure and –by doing this– escape local optimality, tabu search modifies the neighbourhood structure of each solution as the search progresses. The solutions admitted to $N(x)$, the neighbourhood of x, are determined through the use of special memory structures. The search now progresses by iteratively moving from a solution x to a solution x' in $N(x)$.

The most important feature of this algorithm is the use a tabu list, that can be a list of the n last solutions found so far, or a list of forbidden moves, or still a list of forbidden structures. Al-Sultan and Al-Fawzan [ASAF99] developed a **tabu search** to the uncapacitated version of the FLP that gives good results for small- and medium-sized problems. T. Crainic and M. Gendreau have followed this approach for the capacitated version of the FLP, [CG02].

2.4.3 Variable Neighborhood Search [VNS]

Another important heuristic that tries to escape local optimality is VNS. When no better solution is found, a wider and wider neighboorhood is considered to prevent the algorithm to get stuck in a local optimal point. First of all, after finding an initial feasible solution x (randomly or greedily), the neighborhood $N_k(x)$ of the point is progressively explored as long as no better solution is found. k is the size of the neighborhood. As soon as a better solution is found, it is selected and the search starts again with this new solution. Hence this algorithm focuses step after step on a variable neighborhood that depends of the last found solution. The VNS is stopped as soon as $k = k_{max}$ and no better solution has been generated.

∗ **P-median problem: tabu search and VNS**

Hansen and Mladenovic proposed in 1997 [HM97] **a VNS** algorithm for the P-median problem, for which k is the number of location changes. A variation of this method, called co-operative VNS, is a master-slave procedure where the master process initiates several VNS threads, each of which randomly chooses a neighborhood to explore. In the first version of VNS, one random solution is generated in the neighborhood N_k before being locally improved although the co-operative master-slave version CNVNS (Co-operative Neighborhood VNS) generates randomly several solutions for each neighborhood instead of one. Then, the master process is reported the improved solution to an overall solution. The improvements of this version are not really obvious and they depend greatly on the number of parallel searches.

To enhance the performance of TS or VNS, a fairly high-performance heuristic move for the P-median problem is the *Fast Interchange*, [Whi83] and [HM97]. This heuristic move changes at each step the location of one facility. Hence at most m-p facilities are tested for each main iteration. At each iteration, the heuristic measures the benefit of adding one ($goin$) of the closed facilities via a procedure, called Move, that is launched to find the best deletion. Two variables are used : w and the vector $v = (v(1), ..., v(m))$:

- w measures the change in the objective function value obtained by the best interchange ($w*$ is the best one found so far),
- $v(j)$ gives change in the objective function value obtained by deleting the facility j currently in the solution.

To run this algorithm, the closest $c1(i)$ and second closest facility $c2(i)$ must be known in advance and updated after each interchange move via a procedure of $O(nlogn)$ complexity with a heap structure,[HM97].
For each user i, if the new Euclidean distance $d[i, goin]$ is less than $d[i, c2(i)]$, w is updated as follows:

$$w = w + d[i, goin]) - d[i, c1(i)].$$

Otherwise, the change in the objective function that can be caused by the deletion of the facility c1(i) is updated as follows:

$$v[c1(i)] = v[c1(i)] + \min \{ d(i, goin),\ d[i, c2(i)]] - d[i, c1(i)] \}.$$

Then, the facility to be closed *goout* is given by the index whose v value is the lowest, i.e. by the facility whose deletion results in the smallest increase of cost $g = \min_{i \in I} v[c1(i)]$ and w is updated: $w = w + g$.
If $w < w^*$, the interchange is performed. After analysing all closed facilities, the algorithm is stopped if any improvement is done.

∗ **P-center problem: tabu search and VNS**

The 1-interchange descent move was explored and adapted from the P-median problem to the P-center problem [MLH00]: one facility (out of p) belonging to the current solution is replaced by another not belonging to the solution. The initialization step consists of using a random permutation of the facility as a current solution, and find from this initial solution the closest and second closest facility for each user i. Then, iteratively, each of the remaining candidate facility is tested to check whether it is profitable to insert it. In this case, the best facility to delete is searched such as the longest distance between this facility and the users assigned to it is the longest one.

A VNS method is also implemented for the P-center problem [MLH00]: the algorithm remains in the same solution until another better solution is found. The distance between two solutions equals to the number of different facilities. This metric function allows to define a neighborhood of the current solution x: $x' \in N_k(x) \iff \rho(x', x) = k$. As the number of solutions with the same objective value can be high in the P-center problem, the VNS move to a solution with a better or equal value to avoid exploring a solution with maybe a better structure. The local search is done by applying the interchange algorithm.
The computation results of this heuristic methods show that VNS is average better than Tabu Search, although Tabu Search performs slightly better for small p.

2.4.4 Two stage construction solution

Based on the Fast-Interchange heuristic [Whi83], the metaheuristic developed by Rosing and ReVelle [RR96] for solving the P-median problem gives good results, even against Tabu Search [RRR$^+$99]. The first layer of this metaheuristic is to delimit a Concentration Set (CS) via a specific interchange heuristic. The second layer of this method is to determine the best solution in the CS.

Stage one: finding the CS
To denote solutions that can not be improved anymore via a Vertex Substitution Heuristic VSH (as the Fast-Interchange heuristic), the term SPP (for Stable Partitioning Pattern) is preferred to that of local or suboptimal solution since termination of the algorithm is dependent upon the stopping rule of the heuristic and totally unrelated to the gradient of the objective function.
In each problem each SPP differs from each other SPP in having at least two nodes chosen as facilities which are not so chosen in the other SPP by definition of the VSH procedure. The CS is determined by selecting a restricted set of nodes used by a selection of the SPPs found so far.
A statistical table was drawn to answer the question of how many non-optimal SPPs must be inspected to ensure at least one optimal solution is contained in the CS and

the amount of information (number of potential facility nodes) this results in. From 200 heuristics solutions of several instances with $n = 50\,to\,500$ and $p = 5\,to\,50$, it is quite clear that for all instances such that $p \geq 10$, for each of the two best SPPs, between 80% and 100% of the node of each SPP considered lonely are the same as those of the optimal solution and if we consider the nodes opened by the two best SPPs, 90% to 100% of them are those contained in the optimal solution.
No median and very few arithmetic means exceed five. This indicates that in the majority of cases examining five non-optimal solutions will provide a CS containing an optimal set of facilities.

Stage two: selection from the CS
The CS constitutes the restricted set of potential facility sites, built from the nodes that appear in the five best SPPs. The starting Integer Linear Program ILP is restricted to these nodes before being solved. A smaller problem can be formulated based on a strong assumption: the nodes that appear in the solution set of all kept heuristics solutions indicate facilities (CS_o) that are open in the optimal solution although the nodes (CS_f) that appears at least once but not in all kept SPPs are facilities that may or may not be open in the optimal solution. Additional restrictions are performed in the case of P-median problem based on the unlimited capacity of the facilities. The biggest problem restricted to these nodes was solved in 0.7 second instead of about 2700 seconds. Once a solution was found, this solution can be better than the SPPs found so far. However this solution may not be stable in a sense of a VSH and this procedure can be re-applied to this solution before running again the restricted ILP to the new best 5 SPPs.

Designed primarily to provide good-quality solutions, the effectiveness of this method are better than that of tabu search [RRR+99] as this method found in 80 cases the exact optimal solution and in the other 20 cases provided a better solution than TS did. However the computational time can be really lower or really higher than that of TS: the high times are due to the resolution of the reduced problem via Cplex. This suggests that a different exact method or a relaxed solving method might improve the efficiency of HC.

This method combines heuristic and exact methods to try to go beyond local optima as Variable Neighborhood Search or Tabu Search for instance. However this method seems to go beyond typical heuristics by combining an exact procedure. This method presents the property to be designed as a generic method that can be applied to other problems as the P-median one. Any use of this method requires a statistical analysis to evaluate the values of the parameters such as the number of kept SPPs and the number of SPPs to get before applying the exact procedure.

2.4.5 The 3-phased algorithm

The 3-phased algorithm developed by Perl [Per83] for the location routing problem could be used relevantly for the location/assignment problem of servers and clients. This method finds first minimum cost routes. The second step consists of opening the facilities that both respect the routes defined before and minimize the total cost. The third phase attempts to improve this solution by moving customers to other facilities and re-solve the routing problem. Perl's algorithm iterates between the second and

the third phase until a stopping condition is met.

The following derived approach is suggested for our problem:

First, the additional connections for each client to the closest site could be chosen so that the overall unbalance is decreased, regardless of the status of the site. This first phase gives directly the set of sites to be opened. The second step attempts to close sites and re-assign the areas to the left open sites to improve the overall cost. The third step attempts to improve the balance objective by opening sites. This approach iterates again between the second and the third step until a stopping condition is met, as a maximal number of iterations or a satisfying number of efficient solutions.

2.4.6 Scatter search & path-relinking

∗ **Scatter search [SS]**

Both aspects, mono- and multi-objectives problem, are discussed in this section. Scatter search, introduced in 1977 by Glover [SCF06], was used as a heuristic for integer programming. This search orients the exploration relatively to a set of good solutions (P). This set is usually generated by prior solving efforts. Then, a subset of good solutions ($RefSet$) is selected and a solution combination method is performed to create one or more combined solution vectors ($NewSubset$). Two important books, [LM03] and [GLM03], provide a complete prospect of this meta-heuristic.

Several difficulties arise from this approach:

- With a poor selection of the reference solutions ($RefSet$), any procedure to perform combination or improvement will most likely be unable to generate new good solutions ($NewSubset$). In this perspective, integrating as soon as possible the new good combined solutions in the $RefSet$ could lead to faster convergence. However updating this set too early could result in rejecting too early interesting combinations without considering them.

- Besides, it is important to maintain diversity in the $RefSet$ to prevent this set from reaching a steady state too early. To withdraw this drawback, this set should contain solutions of high quality and solutions with a high diversity.

∗ **Path Relinking [PR]**

Path relinking method extends scatter search to the neighborhood space search. This method generates new solutions by exploring trajectories that connect high-quality solutions. The final solution on the generated path is called guiding solution. Such a method tries to incorporate attributes of high quality solutions in intermediate solutions. The moves introduced in the path-relinking method consider the solution structure to build intermediate solutions with a "good attribute composition". A guiding solution may not be better than the starting and ending solutions but may offer an access to new better solutions: hence it is important to scrutinize the neighborhood of the intermediate solutions. However it is generally not efficient to try improving each intermediate solution as they may be too close. On the contrary, a parameter $StepMin$ could be used to allow neighborhood search only every $StepMin$ steps of the relinking process.

Zhang and Lai [YIG06] have implemented path relinking strategy with a genetic algorithm for a multiple-level warehouse layout problem, in a paper accepted in June

2004. Two different path relinking methods for this problem have been implemented to result in only feasible solutions on the path on terms of assignment constraint.

The first PR technique 2.4.1 moves the type indices of the initial solution according to their position in the guiding position.

Algorithm 2.4.1 Algorithm of position-based PR from x_{start} to x_{end}

Initialization:

Set $j = 1$, $x_1 = 0$, $x_2 = 0$, and $x_{middle} = x_{start}$.

Basic Step: iteration k

1. Find i_k in x_{middle} subject to $i_k = \mathbb{R}_j$, where i_k is the i-th component of x_{middle} and $\mathbb{R}_j$ is the j-th component of x_{end}.
   a) If $k = j$, go to step 3.
   b) Otherwise, exchange positions of i_k and $\mathbb{R}_j$. in x_{middle}, and obtain a new middle solution.
2. Evaluate the new middle solution. If it is better than either x_1 or x_2, update x_1 or x_2.
3. Let $j = j+1$.
   a) If $j = n$, output the two best solutions x_1 and x_2 on the path, and stop.
   b) Otherwise go to step 1.

Example 2.4.1: Let $x_{start} = (1, 2, 3, 4)$ be the initial solution and $x_{end} = (4, 3, 1, 2)$ the guiding solution. The following table sums up the run of the position-based PR procedure:

sequence-based PR			
1	2	3	4
4	2	3	1
4	3	2	1
4	3	1	2

The second PR technique 2.4.2 moves two adjacent types of the last intermediate solution according to their sequence in the guiding solution.

Example 2.4.2: Let $x_{start} = (1, 2, 3, 4)$ be the initial solution and $x_{end} = (4, 3, 1, 2)$ the guiding solution. The following table sums up the run of the sequence-based PR procedure:

position-based PR			
1	2	3	4
1	3	2	4
1	3	4	2
3	1	4	2
3	4	1	2
4	3	1	2

Algorithm 2.4.2 Algorithm of sequence-based PR from x_{start} to x_{end}

Initialization:
Set $j = 1$, $x_1 = 0$, $x_2 = 0$, and $x_{middle} = x_{start}$.

Basic Step: iteration k

1. Find $\mathbb{R}_{k1}$, $\mathbb{R}_{k2}$ in x_2 subject to $\mathbb{R}_{k1} = i_j$, $\mathbb{R}_{k2} = i_j + 1$, where i_j and i_{j+1} are type indices in x_{middle}.
   a) If $k_2 > k_1$, go to step 4.
   b) Otherwise, a new solution x_{middle} is obtained by exchanging i_j and i_{j+1} in x_{middle}.
2. Evaluate the new solution. If it is better than either x_1 or x_2, update x_1 or x_2.
3. Let $j = j+1$.
   a) If $j = n$, gp to the next step.
   b) Otherwise go to step 1.
4. Check whether x_{middle} is equal to the reference solution.
   a) If not, set $j = 1$, and go to step 1.
   b) Otherwise, output the two best solutions on the path, and stop.

The first procedure leads to at most $n - 2$ intermediate solutions although the second one results to at most $n \times (n-1)/2 - 1$ intermediate solutions.

Then, the intermediate solutions could be unfeasible and need to be repaired before being improved through a local search. An interesting repair procedure is developed for the biobjective (0-1)-knapsack problem by Carlos Da Silva [SCF06] to repair and improve the unfeasible solutions on the path. This procedure is based on the well-known ratio c_j^i/w_j.

Still more interesting, Gandibleux, Morita and Katoh [GMK04] have sampled a subset of supported solutions for the initial population before using a crossover operator that memorizes the previous assignment to update a frequency wheel used in the crossover procedure. The path-relinking has been used to explore more aggressively the neighborhood of the non dominated set given through an evolution process.

Quite recently, scatter search approach was extended to a multiple-objective $P-$facility location problem [GMJM02]. This scatter search combines a predefined number of selected solutions from a reference set, keeps the facilities in common among all these solutions and add iteratively an additional facility until the number of open facilities reaches P. This combination method seems to behave similarly as the two stage construction method.

2.4.7 Lagrangean relaxation-based heuristic

This method is a widely used technique for combinatorial problems: it is based on the relaxation on constraints that allows to shift from a NP-hard problem to a polynomial problem. The dualization of the side constraints results in an easier problem to solve that provides a bound of the initial problem and often the optimal solution.

Without loss of generality, let consider the following problem:

$$(P)\quad \begin{array}{ll} & v = \min cx \\ \text{s.t.} & Ax \leq b, \\ & Dx \leq e, \\ & x \in \{0,1\}^n \end{array}$$

The corresponding relaxation problem is given by moving a set of constraints to the objective function:

$$(LR_\lambda)\quad \begin{array}{ll} & v_\lambda = \min cx + \lambda(Ax - b) \\ \text{s.t.} & Dx \leq e \\ & x \in \{0,1\}^n \end{array}$$

For $\lambda \geq 0$, we have a lower bound of the optimal solution of (P). By solving (LR_λ) for any $\lambda \geq 0$, we obtain a lower bound of (P). For an equality constraint, λ is unrestricted in sign. Hence, the problem that arises is how to determine λ to obtain the best lower bound for (P). As $Ax - b \leq 0$, the best choice for λ is given by the problem (D) $v_D = \max_{\lambda \geq 0} v_\lambda$, called the Lagrangean dual of (P).
If the set of feasible solutions to (LR_λ) is finite, the set $\{x|.Dx \leq e, x \in \{0,1\}^n\}$ can be expressed as $X = x^t, t = 1, ..., T$. By this way, (D) can be easily expressed as a linear problem:

$$(\overline{D})\quad \begin{array}{ll} & v_D = \max w \\ \text{s.t.} & w \leq cx^t + \lambda(Ax^t - b), t = 1, ..., T \\ & \lambda \geq 0 \end{array}$$

Another dualization of $(\overline{D})$ provides an equivalent problem to our initial problem (P) if ξ_t is integer:

$$(\overline{P})\quad \begin{array}{ll} & v_D = min \sum_{t=\lambda}^{T} \xi_t cx^t \\ \text{s.t.} & \sum_{t=\lambda}^{T} \xi_t Ax^t \leq b \\ & \sum_{t=\lambda}^{T} \xi_t = 1 \\ & \xi_t \geq 0, \{t = 1, ..., T\} \end{array}$$

Either $(\overline{D})$ or $(\overline{P})$ can lead to find the optimal solution of (P). Up to date, the experience seems to indicate that the subgradient optimization related to $(\overline{D})$ is more effective than column-generation-based techniques related to $(\overline{P})$.

Definition 9. *An m-vector γ is a subgradient of v_λ if $v_\lambda \leq v_{\overline{\lambda}} + (\lambda - \overline{\lambda}\gamma)$ for all λ.*

The subgradient method permits the convergence of v^k_{lambda} to v_D if $\theta_k \to 0$ (called the step size) and $\sum_k \theta_k \to +\infty$. The most widely used step sized is : $\theta_k = \frac{\alpha_k(v^* - v^*_\lambda}{\|\gamma_k\|^2}$ with $0 < \alpha_k < 2$ and v^* is an upper bound of v_{lambda} that can be obtained by solving the initial problem (P) via an heuristic.

For location problems, the strategy used by Bazaraa and Sherali [RB81] to manage the step size parameter yields one of the fastest convergence rates. A three stage method [Gal93a] performs well for the Uncapacitated Facility Location $(UFLP)$ problem, composed of 3 stages:

- primal-dual algorithm
- subgradient optimization to solve a Lagrangean dual

- B&B algorithm

$$
\begin{array}{lll}
 & \min \sum_{j=1}^{J} f_j y_j + \sum_{i=1}^{I}\sum_{j=1}^{J} c_{ij}x_{ij} & \\
\text{s.t.} & \sum_{j=1}^{J} x_{ij} \geq 1 & \forall i \in \{1...I\} \quad (15) \\
(UFLP) & & \\
 & \sum_{j=1}^{J} \leq p & (16) \\
 & x_{ij} - y_j \leq 0 & \forall i \in \{1...I\},\ \forall j \in \{1...J\} \quad (17) \\
 & y_j \in \{0,1\}, x_{ij} \in \{0,1\} & \forall i \in \{1...I\},\ \forall j \in \{1...J\} \quad (18)
\end{array}
$$

The$(UFLP)$ can be generalized to the Simple Plant Location problem $(SPFL)$ with $p = |J|$. Hence the constraint (16) becomes redundant and can be removed. The$(UFLP)$ can be generalized to the Simple Plant Location problem $(SPFL)$ with $p = |J|$. Hence the constraint (16) becomes redundant and can be removed. The algorithm (2.4.3) sums up how the $(UFLP)$ (and by consequence the $(SPFL)$) can be solved via the Lagrangean-relaxation tool through this formulation:

$$
\begin{array}{lll}
 & v_\lambda(UFLP] & = \min \left\{ \sum_{j=1}^{J} f_j y_j + \sum_{i=1}^{I}\sum_{j=1}^{J} c_{ij}x_{ij} + \sum_{i=1}^{I} \lambda_i (1 - \sum_{j \in J} x_{ij} \right\} \\
(UFLP\backslash R) & & = \min \sum_{i=1}^{I} \lambda_i - \sum_{j=1}^{J} \left(\sum_{i=1}^{I} (\lambda_i - c_{ij}) x_{ij} - f_j y_j \right) \\
 & \text{s.t.} & (16) \text{ to } (18)
\end{array}
$$

Algorithm 2.4.3 Lagrangean relaxation

Primal-dual algorithm

1. Generate an initial primal solution through a vertex addition heuristic.
2. Define the values of the initial dual variables by the primal solution.
3. Perform a dual ascent procedure to find a solution that allows the identification of a reduced set of vertices.
4. Perform a substitution vertex heuristic by selecting the vertices identified just before.
5. If better bounds are given by either the primal or dual, check *if* the lower and upper bounds coincide, then an optimal solution is found *otherwise* go to step 3.
6. Otherwise go to the next stage.

Relaxation-Lagrangean based algorithm

1. The next Lagrangean relaxation of $(UFLP)$, $(UFLP\backslash R)$, is considered for which $\lambda \geq 0$ is the vector of Lagrangean multipliers with respect to the constraint (2).
2. Let define $s_j(\lambda) = \sum_{i \in I} max(0, \lambda_i - c_{ij})$
3. The optimal y_j are obtained by solving the reduced problem $v_\lambda(UFLP) = \max \sum_{j \in J} s_j(\lambda) y_j$ s.t. $1 \leq \sum_{j \in J} y_j \leq p$, $y_j \in \{0,1\}$, $j \in J$.
4. Update the step size parameter α_k by a function that approaches the positive half of a normal curve
5. *In the case of a duality gap* in the second stage, go to next stage *otherwise an optimal solution is found and stop.*

B&B algorithm

1. Compute a B&B that uses information provided by the previous stage.

By this meta-heuristic, the larged-sized UFLPs were solved without using the

B&B stage up to the size 200 customers $\times$ 200 potential facilities in less than 180 seconds for small value of p and up to 2 hours for $p = |J|$.

The Lagrangean relaxation-based heuristic method has also led to relevant results for capacitated location problems. This method can be used to locate a given number (that is already known) of facilities and warehouses subject to capacity constraints to satisfy the demand of each customer. This method is also valid for multi-commodity version of this problem in which, for a given customer, a different level of demand exists for each commodity [PJ98].
A lower bound is given by solving alternatively two problems (the first one deals with the warehouse allocation and the second one with the facility allocation).
Then, the previous solution is used to update the best lower bound and generate a feasible solution after a subgradient optimization procedure:

- First, make a list of ratio priority according to the shipping cost for each pair,
- Assign each customer to a warehouse or more if the capacity is not enough high,
- Repeat these two steps between the warehouses and the facilities.

The heuristic was tested with small- and medium-sized problems and provided results of good quality (less than 3% higher than the optimal solution) with a very reduced computational time (100 to 200 times faster than the exact procedure).

2.4.8 MC-UFLP algorithm

The Multi-Commodity Uncapacitated Facility Location Problem presents some interesting characteristics for our problem. In this algorithm [Sin05], it is supposed that the maximum number of allowable commodities in any facility's configuration equals to the number of commodities k.

- Solve the LP relaxation on the Balinski IP formulation [Bal66] (extended to MC-UFLP) where the variables x, y are not constrained as non-negative integers but only as non-negative real numbers. The optimal solution (x, y) is memorized and the objective value function OPT_{MCFL} is kept as a lower bound.
- Filter the solution such that each customer is only served by the facilities which are closed to it via the filtering technique of Lin and Vitter [LV92].
- To round the fractional solution (x, y), a set of representative customers is defined and the demand of all other customers is transferred to its representative customer. This set is built such that no two representatives of the same commodity are served by the same facility. This new solution is viewed as the solution of a k-set cover problem between the fractionally open facilities and the representative customers. Let create a set (i, σ) with cost $f(\sigma)$ for every facility configuration pair (i, σ). Let z_i^σ be 1 if set (i, σ) is included in our solution. Our universe consists of all clients in R, and a client $j \in R$ is included in a set (i, σ) if and only if $\bar{x_{ij}}^\sigma$. This problem can be rounded to an integer solution using the result of the following $k-$set cover formulation problem:

$$min \sum_{i,\sigma} f_i(\sigma) z_i^\sigma$$

$$\text{s.t.} \sum_{(i,\sigma):\bar{x}_{ij}^\sigma} z_i^\sigma$$

Such a formulation minimizes the total service cost such that for each representative customer one configuration (i, σ) is kept if there is at least one $x_{ij}^\sigma > 0$, i.e. the representative customer needs the commodity given by the configuration (i, σ).

- $y = \hat{z}$ and the other customers are assigned to the open facities according to the variables $\hat{x}_{ij}^\sigma$
- Finally, the other customers are assigned to the facility which serves its representative customer.

Chapter 3

Multiple-objective Facility Location Problem investigation

As FLP represent real situations, it is not astonishing to find such a high number of papers that consider simultaneously several objectives for the same problem. A survey that can not be overlooked, [NPRC05], presents a complete chapter dedicated to Multiple-objective FLP [MO-FLP] as this book [EG02] mentions all the aspects concerning multiple objective problems. A section of the article [KD04] dedicated to MO-FLP is also reported. A comprehensive reading of these two collections of references seems to prove that our problem has not been deeply studied. Nevertheless this section reports approaches that appear interesting for our study.

3.1 Objective scalarization-based method

Let introduce few notations for the Median and center objective function:

$$\min_{X_p \subseteq P(G), |X_p|=p} \left(\sum_{v \in V} w_v d(v, X_p),\ \max_{v \in V} u_v d(v, X_p) \right)$$

where $N = (G, l)$ denote a network with underlying graph $G = (V, E)$ whose the sef of nodes is denoted V and the set of edges E and p facilities have to be located in any points of (G,l). The distance from a node to a set of p points, $X_p \subseteq P(G)$, is given by $d(v, X_p) = min_{x \in X_p} \overline{d}(v, x)$.

One way to deal with this bi-objective problem is to perform a scalarization of the objective functions that can be viewed as a trade-off of the two objectives, [Hal76]:
$\min_{X_p \subseteq P(G), |X_p|=p} \left(\lambda \sum_{v \in V} w_v d(v, X_p) + (1 - \lambda) \max_{v \in V} u_v d(v, X_p) \right)$
with $\lambda \in [0, 1]$.
This first approach does not ensure to provide a complete set of efficient solutions due to the non-convexity of the objective functions of this combinatorial problem.

Many people have approached the multiple objective problem through the use of weighting factors to scalarize the objective functions into one objective function. Most of the times, not all the supported efficient solutions are enumerated as only a limited set of weighting factors are used, [MHM95] and [MM00]. These case studies were solved by solver softwares as Hyper-Lindo or Lingo.

3.2 $\epsilon-$constraint method

However an additional approach is to consider one objective function with respect to a restriction of the other functions, [Hal80], named $\epsilon-$constraint method. The MOP can be substituted by a sequence of single-objective problems [Ehr05a], denoted $P_{\epsilon-constraint}$:

$$min \ \ f_j(x) \tag{3.1}$$

$$f_k(x) \leq \epsilon_k, \ \ \forall k \in \{1,.,j-1\} \cup \{j+1,..,p\} \tag{3.2}$$

$$x \in X \tag{3.3}$$

Such a method can lead to good performance especially when one of the objective functions is very costly to improve [LV92]. In our case, preliminary computational results have confirmed that the rate function is far much harder to solve than the cost function. The overall algorithm can be summed up as follows:

Algorithm 3.2.1 $\epsilon-$constraint algorithm

Initialization:

Solve the single-objective function f_1 subject to $x \in X$ to determine f_{1I}.
Solve Pl_{F_2/F_1} with $f_1 = f_{1I}$ to determine f_{2N}.
Fix $\epsilon = f_{2N}$ - step.

Basic Step: iteration k

1. Solve the problem $P_{\epsilon-constraint}$ problem to find the following non-dominated point if it exists.
2. Check if a new non-dominated point is found:
   a) If yes, add it to a list that stores the efficient solutions and then decrease the ϵ value by a step : $\epsilon = \epsilon - step$.
   b) Otherwise the stop condition is met.
3. Go to Step 1.

The strength of this method is to focus on a mono-objective optimization problem that might be easier to solve. As shown by first computational results , the cost function is far easier to solve than the balance function. Conversely, determining the complete set of non-dominated solutions requires to solve many times a more and more constrained problem.

Let $F_1 = (f_1^I, f_2^N)$ (respectively $F_2 = (f_1^N, f_2^I)$) be the coordinates of two different non-dominated solutions. Then, minimizing the Tchebycheff norm of the bi-objective problem to the local ideal point can provide a new non-dominated point.

$$min \ \ max \left(\frac{|f_1 - f_{1I}|}{|f_{1N} - f_{1I}|}, \frac{|f_2 - f_{2I}|}{|f_{2N} - f_{2I}|} \right) \tag{3.4}$$

$$\text{subject to } x \in X \tag{3.5}$$

The methods based on Tchebycheff norms select outcomes, $T = (f_{1T}, f_{2T})$, with minimum Thebychev distance from the local ideal point. The initial (real) Ideal point is obtained from the extremal non-dominated points that gives the best solution

according to each objective function. Such a procedure increases the complexity of our problem as it results in a quadratic objective function. Even if it can be linearized, this approach fits well only problem with easy objective functions as cost function: our problem presents a balance function far more difficult to solve than a cost function.

Theorem 1. *If a new point T is found by solving the problem above, there exists no other point that can improve the two objectives of the point T.*

Proof:
Let us suppose there exists a point $\overline{X} = (\overline{f_1}, \overline{f_2})$ that is better on the two objectives than the point T: $|\overline{f_1} - f_{1I}| < |f_{1T} - f_{1I}|$ and $|\overline{f_1} - f_{2I}| < |f_{2T} - f_{2I}|$.
Then, $min\ max\{|\overline{f_1} - f_{1I}|, \overline{f_1} - f_{2I}|\} < min\ max\{|f_{1T} - f_{1I}|, |f_{2T} - f_{2I}|\}$
This is contradictory with the definition of the point T.

3.3 Scalarization-based heuristic

The method developped by [FP03] to solve an UFLP is based on the resolution of two nested subproblems: the plant selection (PS) subproblem and the allocation (A) subproblem. The goal is to minimise a set of cost functions whose value depends on the opening cost of each facility and the allocation cost between each customer. To enhance the interactivity of the association of the (PS_y) and (A_x subproblems, the enumeration of the potential sets of open sites, denoted I, is restricted by tight bounds.
For a given state I, solving the (PS) subproblem is straightforward. The set of efficient allocations for $A_x(I)$ is given by means of the efficient allocation of each client, that is a subproblem easy to solve. Then, to obtain all the non-dominated solutions to the allocation (A) problem, it is just necessary to find all the non-dominated length paths in the network.
Upper bounds are given by solving scalarized plant location problem with different scalar factors, exactly or approximately (in that case, larger upper bounds are obtained but are still valid for the elimination tests). Lower bounds are given by using the uncapacitated structure of this problem.
Step after step, the overall algorithm focuses on an increasing number of open plants. Adding one plant to the current state I relies on the number of uses of each open plant of the non-dominated points found so far: to improve the efficiency of the lower bound rules, the choice of the additional plant is made according to the decreasing number of uses.
The computational results show that the elimination test decreases by five the required time on small- and medium-sized instances (10 to 20 facilities and 20 to 50 customers). However the required space to store the search tree information is enormous (about 500MB for 20×50 instances).

Another paper [BR98] treats the bi-objective UFLP via a method based on scalarization of the objectives. It provides interesting results about the integer friendliness of the problem that consists of both minimizing the total cost and maximizing the demand served. A weighting method is used to provide a partial efficient solution set. Each weighting problem is solved via a LP relaxation of the allocation variables: the proportion of fractional solutions is really low for small-instances (less than 3%)

but deteriorates for larger problem sizes as for the (100×500) instances, 46 % of the solutions are fractional. However for many of these solutions, very little branching are required to find an optimal solution.

3.4 Exact solving of a Hierarchical bi-objective Location problem

Very few articles are devoted to capacitated location problems that consider the equilibrium of loading between the facilities. Among them, a Hierarchical Location of Perinatal Facilities (HFLP) deals with a constraint of balance by Galvao et al [REBY06]. Three different level services are available: 1, 2 and 3. In this model, the mothers-to-be go to a basic unit that offers only service level 1, and can give birth in a maternity home (offers level services 1 and 2) or a neonatal clinic (offers level services 1,2 and 3) for the mothers-to-be has been told to go directly to a neonatal clinic or if any complication implies to transfer her from a maternity home to a clinic. The mono-objective version of this problem is to minimize the total travelling cost, as there is not fixed cost associated to the opening of a structure. Only the level service 3 is capacitated and the capacity limitation, Q, is the same for all clinics. Travel to service level 1 is always to the nearest facility, but not that for service level 3 that is capacitated and that for service level 2 that generate referrals for a fixed proportion θ to level service 3. In case, the neonatal clinics can not absorb the total demand, the existing capacity will be assigned to among all communities such as the same proportion of mothers-to-be of each community will have to seek service outside the Government system. Otherwise, the author worked on improving the "unevenness" of the solutions: the two-objective version of this problem considers simultaneously the total travelling cost and the unbalance by minimizing the maximum deviation of facility usage from average, Z_U.

Hence, a Lagrangean heuristic has been used to solve the starting mono-objective problem regardless of the balance. The optimal solutions requires an enormous amount of time with a PIII 1Ghz-processor computer on Cplex 7.5: the most difficult problem was optimally solved in about 89,000 seconds although only 260 seconds were necessary via the Lagrangean heuristic for which the number of iterations were limited to 2000. Nevertheless, the deviation was always greater than 7The applied methods is similar to that of $\epsilon-$constraint. The decision maker is asked to choose a small set of maximum deviation $\{\rho_1, ..., \rho_q\}$ and the procedure solves the mono-objective problem subject to the additional constraint: $Z_U \leq \rho,\ for \rho \in \{\rho_1, ..., \rho_q\}$. As a consequence, this method does not give all non-dominated solutions. Due to a too high computing time, the real studied problem was tried to be solved via Cplex and not via the Lagrangean heuristic used for the mono-objective version of this problem: it was not possible to solve via Ilog CPLEX this problem.

3.5 Evolutionary algorithms

Genetic algorithms were formally introduced in the US by John Holland at University of Michigan in the 1970s. The continuing price/performance improvements of computational systems has made them attractive for some types of optimization. In

particular, genetic algorithms work very well on mixed integer programming, combinatorial problems. They are less susceptible to getting stuck at local optima than gradient search methods.
The vocabulary used is taken from natural genetics, based on Darwin's [Dar58]evolutionary theory. An overall population of individuals or chromosomes is treated and used to find a best solution after several generations.
From now, the notation MOEA will be used instead of Multiple Optimization Evolutionary

This algorithm consists of a mimic of the nature: natural selection and genetic changes in the individuals:

- Natural selection means that the most adapted individuals survive from one generation to the following one.
- Genetic changes can occur either through a crossover between two individuals or through a mutation underwent by an individual. Such variations are usually random.

Several genetic operators and ***data structures*** (to represent the individuals) have been developed by Michalewicz [Mic98], among others. The results of an evolution algorithm depends greatly on the data structures used to implement the individuals. Such a representation has to be convenient to check its feasibility and flexible enough to be modified via any genetic operator.

Any evolution program follows this generic structure:

- A way to create an *initial solution*
- An evolution iterative phase to create a *new generation* at the end of the iteration through an evaluation function to rate each candidate and a set of selection criteria

Solving exactly a mono-objective location-allocation problem is a NP-hard problem. Exact computational results lead to solve only small-sized instances. An interesting comparison [HJJK96] between three different heuristics has been developed in 1996: a random restart method (RR), a two-opt approach (called H4 heuristic of Love and Juel) and a genetic algorithm. These methods were tested on a location-allocation problem in which both the location of n new facilities (NFs) and the allocation of the flow requirements of m existing facilities (EFs) to the NFs are determined so that total transportation costs are minimized.
Given $mEFs$ located at known points a_j with associated flow requirements $w_j, j = 1, ..., m$ and the number of NFs, n, from which the flow requirements of the EFs are satisfied, the location-allocation problem can be formulated as the following nonlinear program:

$$\begin{aligned} & min \sum_{j=1...m} \sum_{i=1...n} \frac{w_{ij}}{d_{(X_j,a_j)}} \\ \text{subject to} \quad & \sum_{i=1...n} w_{ij} = w_j \qquad \forall j = 1, ..., m \\ & w_{ij} > 0 \qquad \forall (i,j) \end{aligned}$$

where the unknown variable locations of the NFs, X_i, $i = 1, ...,$, and the flows from each NF_i to each EF_j, $w_{ij}, i = 1, ..., n, j = 1, ..., m$ are the decision variables to be determined, and $d_{(X_j,a_j)}$ is the distance between NF_i and EF_j.

- Quality
  For the very large problems, the problems 6 (with 250 existing factories and 25 new factories) and 7 (with 500 existing factories and 40 new factories), the GA provides statistically significant better solutions than either RR or H4 [LJ82] : a t-test was performed and showed that the GA provides a solution less than 0.5 percent higher than the best solution although the other methods were about 4 and 7 percent higher.

- Efficiency
  The computational efficiency of the GA was far better than that of the two other methods, about 4 to 5 times less times as their counterparts.

A more comprehensive review of genetic algorithms is performed in the chapter 5.

Chapter 4

Compact summary of the State of Art

The overall results of the sections 2 and 3 are sumed up in the table 4.1. The characteristics of every relevant paper have been gathered in a table that reveals the related problem, whether the problem is a mono- or multi-objective problem and the method(s) developed by the author(s).

FLP investigation															
Reference	General FLP	UFLP	CFLP	$P-$median	$P-$center	Mono-objective	Multi-objective	Exact	Sub-exact*	Lagrangean	Building-heuristic	TS	VNS	GA
[Erl78]		×				×		×						
[Gal93b]		×				×				×				
[Das95]		×				×				×				
[Geo74]		×				×		×		×				
[KH63]	×	×		×	×	×				×				
[CFN77]	×	×		×	×	×				×				
[Mar64]				×		×			×					
[TP68]		×				×			×					
[Glo77]	×					×						×		
[ASAF99]		×				×						×		
[CG02]			×			×						×		
[HM97]				×		×					×		×	
[Whi83]				×		×					×		×	
[MLH00]					×	×					×	×		
[RRR$^+$99]				×		×						×		
[RR96]				×		×					×			
[Per83]			×			×					×			
[RB81]						×				×				
[Gal93a]		×				×		×	×					
[PJ98]			×			×			×	×				
[Sin05]			×			×			×	×				
[Hal76]	×						×	×						
[MHM95]	×						×	×						
[MM00]	×						×	×						
[Hal80]	×						×							
[FP03]		×					×		×					
[BR98]		×					×		×					
[REBY06]		×					×	×						
[ERM89]							×	×						
[Ste82]							×	×						
[HJJK96]		×				×								×
[FIS05]		×	×				×							×
[YIG06]		×				×					×			×
[GMJM02]				×	×	×				×				

Building-methods: scatter search, path-relinking, 2-stage construction, 3-phased algorithm

*** Solving exact subproblem-based heuristic**

Table 4.1: State of the art

Chapter 5

Investigation of Genetic Algorithms

To deal with a NP-hard problem such as ours in a multiobjective context, the genetic algorithms can be considered as a very promising approach. This approach is still too much misregarded as shown by the less rich litterature in relation with GA. Among the papers covered to gather all the tools necessary to build a well-functioning optimization method, three renowned books [Mic98],[CVL02] and [Deb02], were used. Without pretending to be exhaustive, this section gathers important tools in the perspective of implementing a GA for our problem.

5.1 Population size

Population size is one of the most critical choices faced during the development of a GA.
One approach is a varying population size that has been run by Michalewicz and Mulanka, called GAVaPS or Genetic Algorithm with Varying Population Size. When an individual is generated, its life time is assigned when it is evaluated. The reader should notice that the life time for a given individual does not change through the evolution process.
Contrary to usual GAs, the individual are chosen randomly to reproduce. Once the age of an individual exceed the number of generations during which the individual stays alive, he dies and he is not considered any more during the rest of the process. A constant life time assignment would result in an exponential population growth and would give poor performance. A more complex strategy should be used to reinforce the individuals with a good evaluation and tune the size of the population to current stage of the search. The former goal means that an exponential population growth has to be avoided and the population size has to be minimized in order to maintain low computational costs.

5.2 Genetic operators classification

Genetic algorithms (GA) have been successfully used for bi-objective problems with a complexe objective function (as the balance rate function for our problem). The facility location problem is the class of problem from which our problem is derived.

Hence the results for FLP can be derived to the problem we focus on. A huge wide of papers that treat this class of problem is available, as well as two renowned books [NP05] and [HD03] and software systems (among them, the LOLA platform include a wide-range of LFP algorithms [DoMUoK06]). The variety of real situations explains why it is expected to find such a rich literature. Without pretending to be exhaustive, this section presents the GAs that could be used as a solving technique for our study.

5.2.1 Crossover operator

A classical crossover or one-point crossover can not build potential chromosomes. This drawback has led further researches to develop multi-point crossovers. A probability p is used to decide to which offspring a chromosome component is assigned. However the result of such an approach can be better or worse than that of GSA according to the classes of problem, [SJ91] and [Sys89].

5.2.2 Mutation operator

An individual is taken from the population and altered to provide an offspring. The modification of the parent is generally done by changing randomly one decision variabl. This operator is useful to increase the diversity of the population.

5.3 Additional difficulties in Multi-objective context

As exposed below, the multi-objective problems deal with design difficulties that are specific to this class of problem and with a commun difficulty for all multiple-objective heuristics.

Difficulties in algorithm design
Developing a GA can not be totally successful without regarding 3 main design issues:
(1) Fitness assignment
(2) Diversity preservation
(3) Elitism

1. *Fitness assignment*
   The fitness function is used in GA to assess a value to each member of the populations that is needed for comparison between them. Contrary to mono-objective problems for which the fitness and objective functions are often identical, it can not be the same in a multi-objective context. Three main views have been explored:
   - Aggregation of the objectives to a mono-objective function and the aggregation parameters are varied to find a set of efficient solutions,
   - The candidates are alternatively chosen with respect to one objective that can be chosen randomly (for instance, VEGA [GR87]),
   - The candidates are ranked according to their dominance rank and count (for

instance, SPEA [Bet77]).

2. *Diversity preservation*
   A diversified population presents a well-distributed repartition along the search area, with solutions with a low cost, solutions with a low rate and intermediate solutions. Preserving diversity is crucial to ensure a wide set of efficient solutions, [Zit98] and [Zit02]. That is why three main approaches have been developed to assess the density of the neighborhood for a given solution. Then, the lower this density is, the more its chance to be selected increases.
   - This density of a solution can be equal to the sum of the Kernel function evaluation of all the neighbours of this solution, where the Kernel function takes as parameter the distance.
   - Nearest neighbour techniques keep the distance of a solution to its k^{th} nearest solution.
   - Histograms define a third category of density estimators via an hypergrid which is used to count the number of neighbourhoods of a solution in the same grid.

3. *Elitism*
   Another problem could occur through the evolution process: a good solution could be lost due to random effects. To deal with this problem, the procedure selection could be extended to all the current individuals, i.e. the offspring and the old population. Another solution could be to archive the most promising solutions into an external storage file that is used after each updated generation. It could also be possible to combine these two approaches by generating an archive file based on dominance and density criteria.

Difficulties in assessing the quality of non-dominated solutions

In the case of a single objective problem, assessing a heuristic is straightforward as lower the found solution is, the better the evaluation. On the contrary, comparing two sets of efficient solutions seems intractable as the solutions of a set can dominate some of the other set while others may be incomparable or missing. However it is important to define numerical quality measures of efficient solution set approximations. Even if quality measures can not decide between two algorithms, they are necessary to say in which respect the former is better than the latter and conversely. The generator distances can be used to provide the average distance between each point of the efficient solution set approximation and the closest optimal objective vector. Diversity of the approximation set can also be assessed by a quality measure. However there exists no unary quality measure that is able to indicate whether a efficient solution set approximation is better than another one.
Another difficulty appears in case the non-dominated solution set is not known as many typical ϵ-quality measure are not available any more.
To overcome these limitations, a binary quality measure can be used. For instance, a ϵ-quality measure is presented by Eckart Zitler [Zit01]:
Let (S, T) be two set approximations of non-dominated solutions. Then the binary ϵ-quality measure $I_\epsilon(S, T)$ is defined as the minimum $\epsilon \in \Re$ such that any solution

$b \in T$ is ϵ-dominated by at least one solution $a \in S$:

$$I_\epsilon(S,T) = min(\epsilon \in \Re | \forall b \in T, \exists a \in S : a >_\epsilon b)$$

5.4 Mono-objective GA

5.4.1 DCGA

[MW94] Simple Genetic Algorithms (SGA) with a binary representation presents the drawback to provide a limited precision of the solutions to numerical optimization problem. To overcome it, Delta Coding is a modification of SGA, proposed by Whitley, [Whi02]. The main idea of this approach is to treat individuals not as potential solutions to the problem, but rather as additional values which are added to the current potential solution. The DC procedure can be summed up as follows: after a classical GA run (level x), several runs of the mutation process are executed (level δ). The idea of reinitializing the population was developed by Goldberg [MV95]) who investigates the idea of using small population size that is reinitialized each time the GA converges. The new population keeps only the few best individuals from the previous one.

5.4.2 GENOCOP

[Mic98] Genetic algorithm for Numerical Optimization for COnstrained Problems provides a way of handling constraints (especially linear constraints) which lie in two main steps:
1) elimination of the equalities,
2) design of special genetic operators which guarantee to keep any individual feasible
This approach is valid for a problem whose feasible space D is convex and whose constraints are linear. In such a space, for a given variable x_k, there exists a feasible range $< left(k), right(k) >$ where other variables x_i, $(i \neq k)$ remain fixed.

1) Equality constraints can be written as follows:

$$Ax = b \text{ (A is a } p \times q \text{ matrix, } x = (x_1, ..., x_q).$$

So, there are p independent equality equations, i.e. p variables $x_i^1, ..., x_i^p$. can be determined in terms of the other variables. Hence, the problem is rewritten after substitution of the variables, $x_i^1, ..., x_i^p$. As wanted, this new formulation is free of equality constraints.

2) Operators

- **Uniform mutation**
  This operator modifies only one component x_k of an individual to generate an offspring. The modified component is a random value from the range $\langle left(k), right(k) \rangle$.

- **Boundary mutation**
  This operator modifies only one component x_k of an individual to generate an offspring. The modified component is either equal to $left(k)$ or to $right(k)$, with equal probability.

- **Non-uniform mutation**
  This operator modifies only one component x_k of an individual to generate an offspring. The modified component is equal to :

  $$x_{k'} = \begin{cases} x_k + \Delta(t, right(k) - x_k) & \text{if a random binary digit is 0,} \\ x_k + \Delta(t, x_k - left(k)) & \text{if a random binary digit is 1.} \end{cases}$$

  where t is the generation number.

  The function $\Delta(t, y)$ returns a value in range $[0, y]$ such that the probability of the returned value to be closed to 0 increases through the evolution process. An example function that fits this requirement is : $\Delta_{t,y} = y \times r \times (1 - \frac{t}{T})^b$ where T is the maximum generation number.

- **Arithmetical crossover**
  This operator is defined as a linear combination of two vectors x_1 and x_2. The resulting offspring are $x_{1'} = a \times x_1 + (1 - a) \times x_2$ and $x'_2 = (1 - a) \times x_1 + a \times x_2$ where a is a random value of [0,1]. The convexity of D ensures that $x_{1'}$ and $x_{2'}$ are in D.

- **Simple crossover**
  The usual crossover operator is used to generate from two parents two offspring. To avoid producing offspring out of D, the offspring are those that maximize a (lies between 0 and 1) such that:

  $$x'_1 = < x_1, ..., x_k, y_k \times a + x_{k+1} \times (1 - a), ..., y_q \times a + x_q \times (1 - a) >$$

  and

  $$x'_2 = < y_1, ..., y_k, x_{k+1} \times a + y_{k+1} \times (1 - a), ..., x_q \times a + y_q \times (1 - a) >$$

- **Heuristic crossover**
  Contrary to classical crossovers, this operator produces at most one offspring and uses values of the objective function in determining the search direction. The offspring x_3 is obtained as follows: $x3 = r(x2 - x1) + x2$ where x_2 is not worse than x_1 and r is a random value between 0 and 1.

The first two crossover operators contribute to the stability of the system, with much lower standard deviation of the nested solutions. The last crossover operator has showed to contribute to the precision of the solution found. GENOCOP was tested by Michalewicz on 8 problems and led to good results (quality of solution and computational time) but was only tested on continuous and small problems.

5.4.3 GENETIC

[Mic98] GENETIC is another approach to handle problem with constraints, as the transportation problem: two versions are presented by Michalewicz [Mic98].

GENETIC-1 is developed for linear transportation problems and GENETIC-2 for linear and non-linear ones. These two algorithms deal with balanced problems, i.e. the total demand equals to the total supply. It is possible to transform any transportation problem to a balanced transportation problem, [Tah87].
Contrary to GENOCOP, GENETIC GA is built by using the properties of the transportation structure problem.
GENETIC-1, as a GA for linear transportation problems, uses a vector representation for a solution. If we denote n the number of sources and k the number of destinations, any solution is represented by a $k \times n$ component vector where each component is a unique number between 1 and $k \times n$. A procedure is described to build the corresponding feasible solution from this vector. The inversion, mutation and crossover operators are almost straightforward.
GENETIC-2 uses the most natural representation which is a matrix. The genetic operators are defined as follows in this case:
- the mutation operator creates a $p \times q$ submatrix from the elements of the parent. After the mutation, the column and row sums have to remain the same as before. The initialization procedure used for GENETIC-1 is used to build a new submatrix with respect to the column and row sum constraints.
- The crossover operator allows to build two offspring v_3 and v_4 from two parents $v_1 = (v_{ij}^1)$ and $v_2 = (v_{ij}^2)$ as follows:

- Build the matrixes $div = (div_{ij})$ and $rem = (rem_{ij})$ where $(div_{ij}) = (v_{ij}^1 + v_{ij}^2)/2$ and $(rem_{ij}) = (v_{ij}^1 + v_{ij}^2) mod 2$
- Build rem_1 and rem_2 such that $rem = rem_1 + rem_2$, $sourrem_1[i] = sourrem_2[i] = sourrem_1[i]/2$ and $destrem_1[i] = destrem_2[i] = destrem_1[i]/2$
- $v_3 = div + rem_1$ and $v_4 = div + rem_2$

Computational tests shows that GENETIC-2 performs better than GENETIC-1. Moreover, GENETIC-2 is potentially more fruitful for generalization than GENETIC-1, as it is based on a matrix representation and does not depend on the initialization procedure that is heavily based on knowledge of the solution form (except for mutation operator but it could be substituted).

5.5 Multi-objective GA

5.5.1 NSGA

[SD93] A (Elitist) Non-dominated Sorting GA was proposed in 1994. This methods was improved in 2000, called NSGA-II [DAPM00]. This author developed a GA that do not only use an elite-preservation strategy but preserves also diversity.
From a population P_t of size N, an offspring population Q_t of size N is generated, then both of them ($R_t = P_t \cup Q_t$) are used to generate the next population P_{t+1}. A truncation of the population R_t whose size equals to $2N$ is required to reach a population of N individuals. No archive guarantees to keep all non-dominated solutions found so far contrary to other GAs like SPEA. The individuals are kept with respect to a Crowded Tournament Selection Operator $<_c$ described below as long as the size of the new population do not exceed N. Then, the generated population is processed through mutation and crossover operators to form P_{t+1}.

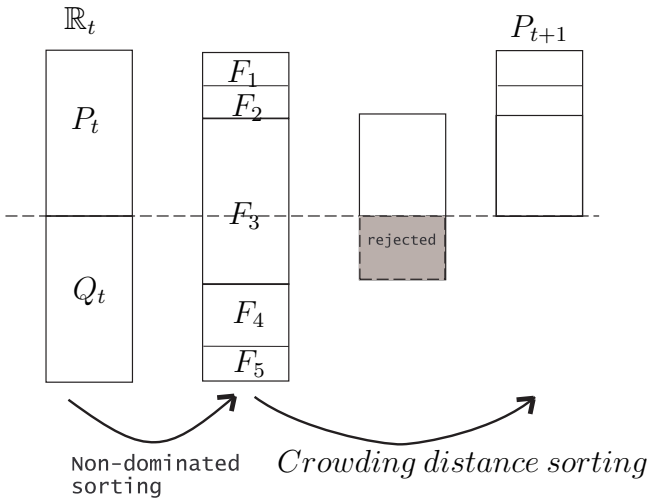


Figure 5.1: NSGA selection procedure

Non-dominated sorting
In comparison with NSGA, the NSGA-II procedure provides a quicker non-dominated sorting procedure in $O(kN^2)$ for a k-objective problem [Deb02]. Two entities are calculated for each solution to sort: n_i, the number of solutions which dominate the solution i, called domination count and S_i, the set of solutions which are dominated by the solution i. At the end of the first iteration, the first front consists of the solution whose the domination count equals to 0. Then, the same procedure is reiterated for each set S_i: the solutions from the sets S_i that are non-dominated by any solution are put in a list P'. Once all the sets S_i are visited, the second non-dominated front is formed by the members of P'. This process continues until there is no remaining set S_i or until the complete non-domination fronts enumerated so far count at least N members.

Crowded distance
This distance is the average side-length of the cuboid formed by the M closest solutions to the candidate i where M is the number of objective functions. The M boundary solutions are assigned an infinite large crowded distance and the others a null one. Sort all the candidates in worse order of $f_m, m = 1...M$ to find the sorted indices vector I_m.
Then for each individual, the j-th indice of I_m is given by:

$$d_{I_j^m} = d_{I_j^m} + \frac{f_m^{I_{j+1}^m} - f_m^{I_{j-1}^m}}{f_m^{max} - f_m^{min}}$$

Then, the crowded distance for each solution j is given by:

$$d_c = mean_{m=1,...,M} d_{I_j^m}$$

As the number of solutions in the first rank population are likely to exceed N at the beginning of the NSGA procedure, such a truncation ensures to choose diverse solutions according to the relative local concentration of the solutions. At the other end of the scale, when approaching the end of the search, the number of solutions

may be greater than N. Hence, this niching procedure is a way to get a good spread among the selected solutions.

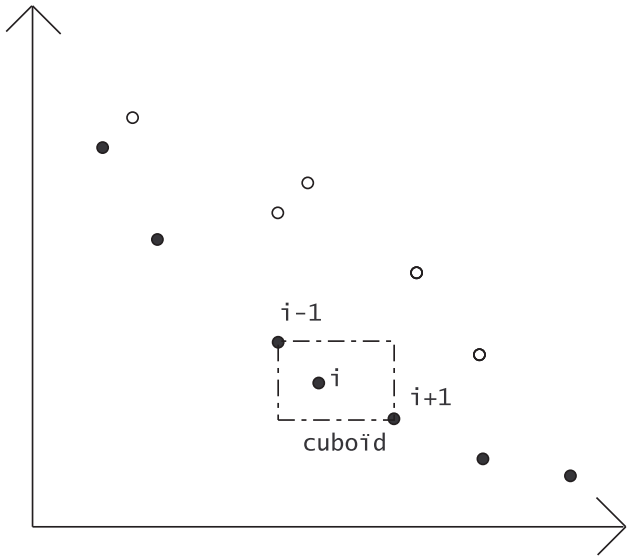


Figure 5.2: Crowding distance calculation

Computational results on a $MinEx$ problem shows that NSGA-II can keep a spread of solutions even after a large number of generations without mutation, even if a mutation operator improves the distribution of the solutions along the front of the non-dominated points.

Crowded Tournament Selection Operator $<_c$
Let R_i be the rank of a solution ($R_i = 1$ for the set of non-dominated solutions and so on).
1. If solution i has a better rank than a solution j, then $R_i < R_j$.
2. If they have the same rank, the solution that has the better crowded distance wins the tournament.

The selection operator $<_c$ was revisited in a problem of optimal design in electrical distribution networks in 2005 by Favuzza [FIS05] to handle constraints. Two crowded comparison operators were compared: CCO1 and CCO2. The former operator considers the constraints as further objectives, in terms of non-domination, as long as they are not satisfied. On the contrary, the latter one considers in first place lower constraints violation, then the rank order and finally the crowding order.

- **CCO1**
  Let consider the following standard bi-objective problem formulation:

$$\begin{aligned} & minf(x) = \{f_1(x), f_2(x)\} \\ s.t. \quad & g(x) = g_1(x), ..., g_s(x) \leq 0 \\ & h(x) = h_1(x), ..., h_p(x) = 0 \end{aligned}$$

where s in the number of inequality constraints and p that of equality constraints. CCO1 considers the following objective function:
$min\ f(x) = \{f(x_1), f(x_2), \alpha_1(g_1(x)), ..., \alpha_s(g_s(x)), \beta_1(h_1(x)), ..., \beta_p(h_p(x))\}$.
The difficulty of this operator comes from the evaluation coefficients of the functions
$\alpha_1, ..., \alpha_s, \beta_1, ..., \beta_p(x)$. The form of $\alpha_j, j = 1...p$ is as follows:

$$\begin{cases} a + b(evalconstr - fixedmember) & \text{if the constraint is violated,} \\ 0 & \text{otherwise.} \end{cases}$$

where $evalconstr$ is the value given by the variable part of the considered constraint and $fixedmember$ is the fixed part of the considered constraint. For any $1 \leq j \leq p$, $\beta_j, j = 1...p$ is similar to that of α_j as a equality constraint can be considered as a pair of inequality constraints. Hence, for each constraint that can be violated, the corresponding coefficients have to be fixed to be meaningful.

- **CCO2**
  To measure the degree of constraint violations, the following CV function is defined for a $k-$objective problem:
  $$CV = \begin{cases} 0 & \text{if there is no constraints violation} \\ \frac{1}{k}(\alpha_1(g_1(x)) + ... + \alpha_s(g_s(x)) + \beta_1(h_1(x)) + ... + \beta_p(h_p(x)) & \text{otherwise} \end{cases}$$

The CCO1 operator appears promising as unfeasible solutions are partially kept and could lead to a faster convergence to the front of non-dominated solutions. The CCO2 operator considers unfeasible solutions only if there are not enough feasible solutions. According to the authors of this paper [FIS05], the new CCO1 operator provides solutions with a better quality and similar (or sometimes better) performance. However, there is no comparison with the $<_c$ operator associated with the rejection of unfeasible offspring and the $<_c$ operator associated with a repairing process of the unfeasible offspring.

5.5.2 NPGA

The Niched-Pareto Genetic Algorithm -[HN93] and [HNG94]- is also based on the non domination concept. Contrary to other GAs explored before (as NSGA), it differs from them in the using of a binary tournament selection.
From a shuffled population P (of parents) and a empty offspring population Q, a NPGA-tournament is performed between two offspring to select a parent p_1. This tournament is reiterated from two other parents to select a second parent p_2. The two offspring, c_1 and c_2, are given through a crossover between p_1 and p_2. Then mutation can be performed on c_1 and c_2. The offspring population is updated $Q = Q \cup \{c1, c2\}$. Then this process is reiterated until all the parents have been explored.
The real advantage of this GA is that no explicit fitness assignment is required. However, the NPGA-tournament depends on two parameters t_{dom} and σ_{share} that greatly influence the performance of this GA.

The NPGA-tournament(i,j,Q) procedure is described below:

Algorithm 5.5.1 NPGA tournament (i,j,Q)

Basic Step: iteration i, j, Q

1. Pick a subpopulation T_{ij} of size t_{dom} from the parent population P.
2. Calculate α_i (respectively α_j)) as the number of solutions in T_{ij} that dominates i (resp. j).
   a) If $\alpha_i > 0$ and $\alpha_j = 0$ (or the contrary), the winner is i and the process is complete (or the contrary).
   b) Otherwise if $|Q| < 2$, the winner is chosen randomly and the process is complete.
   c) Otherwise, the solution that has the lower niching counts (nc_i for solution i) wins the tournament

These two parameters t_{dom} and σ_{share} are calculated to choose the winner by placing successively each of them in the population Q, independently. The niching count is the number of offspring within a σ_{share} distance d_{ik} from i (k represents an offspring):

$$d_{ik} = \sqrt{\sum_{i=1}^{M} (\frac{f_m^{(i)} - f_m^{(k)}}{f_m^{max} - f_{min}^k})^2}$$

where f_m^{max} and f_m^{min} are the maximum and minimum bounds of the m^{th} objective function.

Simulation results show that mutation is required to prevent the population to converge to a unique value of the front of the non-dominated points. However whether a solution is located very close to solution i or a solution is situated at a distance almost equal to σ_{share} is not a concern in this method.

5.5.3 A-SPEA

The Strength Pareto Evolutionary Algorithm has been developed to compete with MOEA and has been compared favourably with other MOEAs in these studies: [Bet77], [BP95], [Zit99] and [Zit00]. In parallel, a comparison with a more recent version of SPEA, A-SPEA [DGD05], is exposed afterwards. The reader can notice that an improved version of SPEA, SPEA2, appears roughly at the same moment but all the improvements of this new version are not necessary useful for our problem.
SPEA algorithms is organized as described by the algorithm 5.5.2.

SPEA2 is built on the same main loop as SPEA. This new version of SPEA improves the SPEA's weaknesses about fitness assignment, and density estimation & archive truncation.

Initial population:
Generating the initial population P_0 is a step that should not be misregarded to prevent the algorithm from converging to a not well-distributed set of non-dominated solutions and a good-quality initial population could enhance the search through the evolution process. A greedy algorithm needs to be built such that a wide-distributed initial population is performed. In case the size of the initial population is not high

Algorithm 5.5.2 SPEA Main Loop

Initialization:

Fix an archive size N, a maximum number of iteration T.

Generate an initial population P_0 and create an empty archive A_0. Set t=0 and $A_0 = \emptyset$.

Main Loop:

1. **Fitness assigment:** Calculate fitness value of individuals in P_t and A_t.
2. **Environmental selection:** Copy all non-dominated individuals in P_t and A_t to A_{t+1}.
3. **Termination:** If $t \geq T$, set $A*$ as the set of decision vectors represented in A_{t+1}.
4. **Mating selection:** Perform binary tournament selection with replacement on A_{t+1} to fill the mating pool.
5. **Variation:** Apply recombination and mutation operators to the mating pool and set P_{t+1} to the resulting population. Increment generation counter $t = t + 1$ and to step 1.

enough, a complementary algorithm would be necessary to complete it.

Fitness assignment:

In SPEA, two different fitness assignment procedures are used to both the archive and the population.

- For i in the archive, $F(i) = S(i) \in [0, 1]$ where $S(i)$ is the number of population members j that are dominated by or equal to i, divided by the population size plus one.
- For j in the population, $F(j)$ is given by summing the strengh values $S(i)$ of all archive members i that dominate or are equal to j, plus one.

As the fitness is to be minimized, this fitness assignment function results in a higher chance for each member of the archive to be selected than any member of the population.However the fitness value is based on a Pareto niching method that preserves diversity. In SPEA2, the fitness value overcomes the situation for which individuals who are dominated by the same archive members are assigned the same fitness value by the calculation of the value R(i). However, this value may fail when most individuals do not dominate each other. That is why the density is considered to avoid this issue by an adaptation of the k^{th} nearest-neighbour method. However this density function is not considered in our case as we deal with a combinatorial problem for which the number of solutions is not as high as that of a continous problem. As our concern is to keep all the non-dominated solutions, the archive size limit is infinite.
The solution is to take into account dominating and dominated indivuals & the density estimation as expressed below:
$F(i) = D(i) + R(i)$
The rank of domination is equal to:

$$R(i) = \sum_{j \in \{(P_t + A_t) \cap j > i\}} S(j).$$

where $S(j) = |\{i \in (P_t + A_t) | i < j\}|$
The density estimation is used to decide between candidates who have the same domination rank, that is an adaptation of the k^{th}-nearest neighbor method : $D(\mathbf{i}) = \frac{1}{\sigma_{\mathbf{i}}^k + 2}$
A common setting for k is $k = \sqrt{N + \overline{N}}$.

Candidate selection :
The archive size remains constant in this method contrary to SPEA. Boundary solutions are not lost through this truncation method that reduces the size of the archive to the maximum allowed size when required.
To update the archive, all the non dominated individuals from the current archive and the population are considered. If the size of the resulting set equals to N, the update is completed. Otherwise there are two cases: either the size is less than N or greater than N. In the latter case, the archive is completed by adding enough remaining individuals that are sorted according to their fitness value. In the former case, the individual that has the minimum distance to another individual is removed, after each iteration. In case they are several individuals with minimum distance, the second smallest distance is considered and so forth.

However this modification from SPEA2 is not kept in our version of SPEA as a archive truncation may prevent from enumerating all the non-dominated solutions, which is an important purpose of our problem.

operators :
A one point crossover operator with a probability equal to 80% was choosen in this section, combined with a mutation probability of 4%. It is likely that only a one point crossover operator will be kept in our case due to the high level of constraints of the solutions. Only statistic evaluations could give the nest crossover and mutation probabilities. Then, a repair algorithm may be useful to repair the solutions if it is decided to keep only feasible solutions in the population. To enhance the quality of the feasible solutions, a local search can be performed in the two directions to allow the SPEA algorithm to be more agressive. In addition, forcing a local search in the direction f_1 (resp f_2) for the current best solution along f_1 (resp f_2) is a relevant way to obtain better extremal solutions.

5.6 Conclusion

The single objective GAs and multiple objective GAs have been summed up in two tables as no relevant direct comparison could be done between this two classes of GAs.

Mono-objective GA		
SGA	Simple GA	
DC	Delta Coding	to increase solution procedure
GAVaPS		Varying Population Size
GENOCOP	Constrainted problem	Suppress equalities constraints Operator generating feasible offspring on a convex search
GENETIC-1	Balanced LP Vector	Vector $\overset{initilizationprocedure}{\rightarrow}$ feasible solution
GENETIC-2	Balanced LP Matrix	Specialized mutation via IP Universal crossover

Multi-objective GA		
NSGA-II	Elitist sorting GA	Fixed sized-population whose individuals belong to several non-dominated fronts Preserve diversity of the individuals among those of the last front
NPGA	Niched-Pareto	Selection operator based on binary tournament
SPEA	Strength Pareto EA	Archive for non dominated candidates
SPEA2		Improvement on fitness assignment & archive truncation

The important thing that appears in this section is the wide set of choices for implementing a GA. This investigation and the quoted applications of these GAs seems to reveal SPEA-2 and NSGA-II perform better than the others. Due to time limitation, only one genetic algorithm can be correctly implemented and adapted to our problem. As no real difference of performance was revealed between these two algorithms, SPEA-2 was chosen because much more efforts and papers related to NSGA-II have been done. SPEA-2 might potentially be improved more than NSGA-II.

Chapter 6

Development on an exact solving method for the model *MMX*

Among the exact solving method for a multi-objective problem presented in the section 3, the method selected for further investigation is that presented by the subsection 3.2: the ϵ-constraint method is a classic approach where $k-1$ objectives are considered as constraints. The reader can find further details in

- Ralph Steuer and R-U Choo. An interactive weighted Tchebychev procedure for multiple objective programming. *Mathematical Programming*, vol. 26, pp. 326-344, 1983.
- M. Ehrgott, *Multicriteria Optimization*, Second Edition Springer, 2005 (section 4.1) [Ehr05b]
- M. Ehrgott and Xavier Gandibleux, Multiple objective combinatorial optimization - a tutorial In *Multiple Programming and Goal-Programming; theory and applications*, T. Tanak, T. Tanino and M. Inuiguchi, Advances in Soft Computing, pp. 3-18. Springer 2003. [LV92]

In the two objective case, the idea is to enumerate the efficient solutions by updating iteratively the value on a constraint on one objective.

6.1 Principles

The MOP can be substituted by a sequence of mono-criterion problems, denoted $P_{\epsilon-const}$:

$$\left[\begin{array}{ll} \min & min\ f_j(x) \\ s.t. & f_k(x) \leq \epsilon_k,\ \forall k \in \{1,.,j-1\} \cup \{j+1,..,p\} \\ & x \in X \end{array}\right]$$

Such a method can lead to good performance especially when one of the objective functions is very costly to improve. In our case, computational results have confirmed that the rate function is far much harder to solve than the cost function, as shown in the table 6.1.

Table 6.1: Comparison of the difficulty to solve each objective function: cost and rate.

Instances	CPUt for minimizing the cost	# CPUt for minimizing the rate
`BiFLP12-5-1.VC.1`	0.09	6
`BiFLP12-5-2.VC.1`	0.7	607
`BiFLP12-5-3.VC.1`	7	436
`BiFLP17-5-1.VC.1`	0.9	1646

Based on the problem $P_{\epsilon-const}$ defined above, the overall algorithm can be summed up by the algorithm 6.1.1.

Algorithm 6.1.1 $\epsilon-$constraint algorithm

Initialization:

Solve the mono-criterion function f_1 subject to $x \in X$ to determine f_{1I}.
Solve $Pl_{2/1}$ with $f_1 = f_{1I}$ to determine f_{2N}.
Fix $\epsilon = f_{2N}$.

Basic Step: iteration k

1. Solve the problem $P_{\epsilon-const}$ to find the following non-dominated point if it exists.
2. Check if a new non-dominated point is found:
   a) If yes, update f_{2N} and then decrease the ϵ value by a step : $\epsilon = f_{2N} - step$.
   b) Otherwise the stop condition is met.
3. Go to Step 1.

Finalization:

Filter the solutions to keep only the non-dominated ones

The filtering step ensures to remove the weakly dominated solutions as illustrated by the next figure 6.1. After determining the point (f_{1I}, f_{2N}), the first iteration gives the solution s_1 and the second iteration the solution s_2. This point dominates the solution s_1 obtained before as the cost of these two solutions are equal but the rate ρ of the solution s_2 is lower than that of s_1.

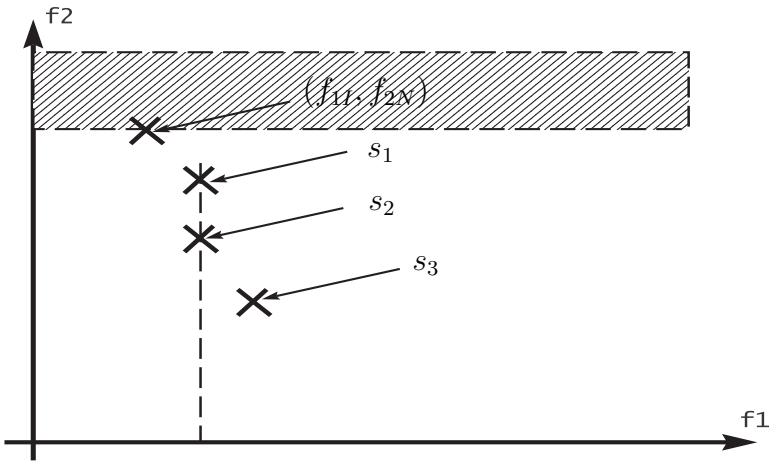


Figure 6.1: ϵ-constraint algorithm

6.2 Adapted algorithm: *EPSIL*

The crucial point of this algorithm **EPSIL** is the evaluation of the step size that influences the performance of the method as well as the number of found non-dominated points. A big step increases the performance of this method although a too wide step can not guarantee to find all non-dominated solutions. However, the idea of using a varying size step with this solving method to sample a set of efficient solutions could be relevant for our purpose. As we have enough knowledge of the range of values for the rate and cost, it is possible to calibrate a set of efficient solutions generated by this method.
Another improvement has been added to this algorithm to increase its performance. The lower bound of the cost is updated after each iteration: it equals to the cost value of the last non-dominated point plus $B_1 = 1$. Indeed we focus on real cost function where all costs are approximated so they can be considered to be integer. Concerning the step B_2, it is not straighforward: we want to find :

$$\delta = Inf\left(\left\{\underbrace{\left|\left\|\frac{\alpha_j}{Q_j} - \frac{\alpha_k}{Q_k}\right\| - \left\|\frac{\alpha_l}{Q_l} - \frac{\alpha_m}{Q_m}\right\|\right|}_{M}\right\} \cap \Re^+\right).$$

$\forall j \in J, \alpha_j = Q_j - \sum_{i=1}^{N} \frac{d_i}{N} x_{ij}$. As $d_i \in N$ and $x_{ij} \in \{0,1\}$, we have $N\alpha_j \in N$.
It results that:

$$\begin{aligned} M &\leq \tfrac{1}{N}\left|\left\|\tfrac{a}{Q_j} - \tfrac{b}{Q_k}\right\| - \left\|\tfrac{c}{Q_l} - \tfrac{d}{Q_m}\right\|\right| && \text{where } (a,b,c,d) \in N^4 \\ &\leq \tfrac{1}{NQ_jQ_kQ_lQ_m}\left|\|a'-b'\| - \|c'-d'\|\right| && \text{where } (a',b',c',d') \in N^4. \end{aligned}$$

As a consequence, the next bound B_2 can be choosen as follows: $\frac{1}{(max_{j\in J}Q_j)^2(max_{j'\in J, j'\neq j}Q_{j'})^2}$.

$$min\ \ f_1(x)$$

$$f_2(x) \leq f_2(x)^* - B_2$$

$$f_1(x) > f_1(x)^* + 1$$

$$x \in X$$

where $f_1(x)^*$ is the cost of the last non-dominated point found so far and $f_2(x)^*$ is the rate of the last non-dominated point found so far.

6.3 Computational results and related conclusions

The used computer presents the following characteristics:
- CPU : Pentium 4, 3.73 GHz
- RAM : 4 Gb
- Operating system : Linux Debian ; kernel 2.6.14.5
- Compiler : gcc without optimizing option

The used solving MILP is the following one:
- Name : ILOG CPLEX
- Version : 9.100
- Specific parameters : default options e m b q
- Using mode : specific implementation in C with call to the CPLEX library
- Compiler : gcc without optimizing option

The model MMX is solved with the method Espil on instances produced as follows:
- the cost of each site is randomly chosen such if a site i has a capacity strictly greater than another site i', the cost of the site i is also strictly greater than that of the site i'
- the cost connection between a site and an area is directly proportional to the distance between these two facilites.

Each bi-objective location problem is summarized by the following characteristics:

biFLP `Na - Ns - Nc . SC . Vx`

where `Na` gives the number of areas, `Ns` that of sites, and `Vx` the version number. `SC` means that the capacity of the sites are all equal to each other, `VC` otherwise. `Nc` gives the number of connections.

For instance, the name *biFLP75-50-2.VC.3* refers to a problem with 75 areas, 50 sites, 2 connections, various capacities, instance number 3.

The results of the biggest instances that could be solved exactly in less than 24 hours are summed up in the tables 6.2, 6.3, 6.4 and 6.5. The first column gives details about the instance that is solved, the second column gives the name of the solving method, the third one provides the number of non-dominated solutions, the fourth column is necessary to gives the time comsumed to solve the problem (in seconds) and the last column informs about the number of branch & bounds.

As shown by these results, the difficulty to solve this bi-objective problem increases quickly with the size of the problem. It was tried to give more time to Cplex to solve bigger instances as the `BiFLP17-5-2.VC.1`. However even 120 hours were not enough to solve bigger instances.

Table 6.2: Computational results for 8-area instances: model MMX.

Instances	Method	# ND points	CPUt	# nodes(B&B)
BiFLP8-5-1.VC.1	Epsil	8	1,71	9 008
BiFLP8-5-2.VC.1	Epsil	10	5,61	30 768
BiFLP8-5-3.VC.1	Epsil	6	1,77	10 857

Table 6.3: Computational results for 10-area instances: model MMX.

Instances	Method	# ND points	CPUt	# nodes(B&B)
BiFLP10-5-1.VC.1	Epsil	3	0,73	3 064
BiFLP10-5-2.VC.1	Epsil	19	2,87	13 319
BiFLP10-5-3.VC.1	Epsil	79	5,5	13 308

Table 6.4: Computational results for 12-area instances: model MMX.

Instances	Method	# ND points	CPUt	# nodes(B&B)
BiFLP12-5-1.VC.1	Epsil	9	35,1	187 806
BiFLP12-5-2.VC.1	Epsil	14	600,1	3 222 633
BiFLP12-5-3.VC.1	Epsil	23	2391	9 056 321

Table 6.5: Computational results for 17-area instances: model MMX.

Instances	Method	# ND points	CPUt	# nodes(B&B)
BiFLP17-5-1.VC.1	Epsil	12	6614	26 486 344

Chapter 7

Methods to build the initial population

7.1 Other formulation of the problem *AVG*

The difficulty to solve this bi-objective problem is above all due to the objective function that minimizes the rate ρ as illustrated by the results of the table 6.1. For this reason, a second model was formulated to solve the bi-objective location problem. Without pretending to be equivalent to the first model, the expected advantage of this alternative model is to provide an easier problem to solve for the MIP solver.

7.1.1 The principle

Starting from the following problem:

$$\left[\begin{array}{ll} \min & \mid p(x) - q(x) \mid \\ & p(x),\ q(x) \geq 0 \end{array}\right]$$

the objective function can be optimized according to the following principle:

$$\begin{array}{l} s^+ \geq 0,\ s^- \geq 0 \\ p(x) - q(x) \ = \ s^+ - s^- \\ \text{and minimize } s^+ + s^- \end{array}$$

Hence, the new problem to solved is

$$\left[\begin{array}{lll} \min & s^+ + s^- & \\ s.t. & p(x) - q(x) & = \quad s^+ - s^- \\ & p(x),\ q(x), s^+, s^- \geq 0 & \end{array}\right]$$

In this model, the difference of loading between each pair of sites (whatever their status) is measured positively and the sum of these differences is minimized (see figure 7.1).

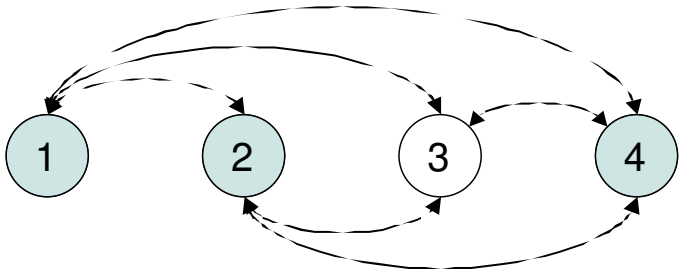


Figure 7.1: Measure of the difference of loading

Using the new approach, the formulation of this problem is expressed as follows:

$$
\left[\begin{array}{lll}
\min \text{Cost} = & \displaystyle\sum_{j=1}^{J} f_j y_j + \sum_{i=1}^{I}\sum_{j=1}^{J} c_{ij} x_{ij} & (1.1) \\
\min \text{Rate} = & \displaystyle\sum_{i=1}^{J(J-1)/2} sp_i + sn_i & (1.2) \\
\text{s.t.} & \displaystyle\sum_{j=1}^{J} x_{ij} = N \qquad \forall i \in \{1,...,I\} & (2.1) \\
& y_j Q_j = \displaystyle\sum_{i=1}^{I} \frac{d_i}{N} x_{ij} + \alpha_j \qquad \forall j \in \{1,...,J\} & (2.2) \\
& \frac{\alpha_j}{Q_j} - \frac{\alpha_{j'}}{Q_{j'}} = sp_{j'-j+J-1} - sn_{j'-j+J-1} \quad \forall (j,j') \in \{1,...,J-1\} & (2.3) \\
& \qquad\qquad \times \{j+1,...,J\} & \\
& y_j \in \{0,1\}, x_{ij} \in \{0,1\}, \alpha_j \geq 0 & (3.1)
\end{array}\right]
$$

The decision variables are respectively y_j and x_{ij} with:

$$y_j = \begin{cases} 1 & \text{if the site } j \text{ is open,} \\ 0 & \text{otherwise.} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if the site } j \text{ is assigned to the area } i, \\ 0 & \text{otherwise.} \end{cases}$$

The constraint (2.1) ensures that each area is connected to N sites. The constraint (2.2) expresses the capacity limit for each site. Besides, such an equation ensures that if a site is closed, no area is connected to this site. The constraint (2.3) is used to evaluate the imbalance rate. The constraints (2.1-2.3) plus (3.1) define the decision space denoted X.

The difference between this proposed model and the model MMX is the balance objective function: this model focuses on minimizing the summation of discrepancy between the level load of all the pairs of sites, whatever their status, although the model MMX minimizes the biggest difference of loading between the open sites. Concerning the rate ρ of the solutions provided by the model AVG, a significant gap may be observed. The main motivation of this model is to develop an easier problem to solve via Cplex. In this case, this modelling could be used to build a part of a promising initial population used by the MOEA developed for the model MMX.

7.1.2 The AVG model solved with the EPSIL method: *EPS*

In the method *EPS*, the model AVG is solved with the method Epsil. As the goal of this part is to build a set of starting solutions to get later the non-dominated solutions, a time limit is used to stop the exploration. Besides, this limit is necessary to control the behavior of the MOEA.

7.1.3 The AVG model solved with a dichotomous reduction of the search space: *EPSDIC*

One important drawback of the method *EPS* is the high number of solutions with a relatively good cost that are generated in comparison with the number of solutions that favour the rate ρ of the AVG model (see the note (NT3/CRE FT-LINA/v060714)). To overcome this drawback, the space search is divided by a dichotomy to direct the search towards the search spaces where no solution has been still generated.

More precisely the method EPSDIC starts with two rate values -ρ_{low} and ρ_{up}- that are used to delimit the search space. Let consider the exploration space $subint(\rho_{low}, \rho_{up})$:

$$\left[\begin{array}{c} subint(\rho_{low}, \rho_{up}) : \\ x \in X \\ \rho_{low} \leq \rho \leq \rho_{up} \end{array}\right]$$

Then, it is is divided at each loop in 2 subintervals $-upperint(\rho_{low}, \rho_{up})$ and $lowerrint(\rho_{low}, \rho_{up})-$ defined as follows:

$$\left[\begin{array}{l} lowerint(\rho_{low}, \rho_{start}) : \\ \quad x \in X \\ \rho_{low} \leq \rho \leq \rho_{low} + \frac{1}{2} \times (\rho_{start} - \rho_{low}) \\ \\ upperint(\rho_{low}, \rho_{start}) : \\ \quad x \in X \\ \rho_{low} + \frac{1}{2} \times (\rho_{start} - \rho_{low}) \leq \rho \leq 1 \times \rho_{start} \end{array}\right]$$

The method *EPSDIC* looks for a solution in each of these subintervals by solving the following problems:

- $P_{lowerint(\rho_{low}, \rho_{start})}$

$$\left\{\begin{array}{ll} \min & Cost \\ s.t. & lowerint(\rho_{low}, \rho_{start}) \end{array}\right\}$$

- $P_{upperint(\rho_{low}, \rho_{start})}$

$$\left\{\begin{array}{ll} \min & Cost \\ s.t. & upperint(\rho_{low}, \rho_{start}) \end{array}\right\}$$

Afterwards, if a solution is found in the interval *lowerint* (ρ_{low}, ρ_{start}) (respectively *upperint* (ρ_{low}, ρ_{start}), this intervall *lowerint* (ρ_{low}, ρ_{start}) (respectively *upperint* (ρ_{low}, ρ_{start}) is divided into two subintervals and the same procedure is launched again in each of these two subintervals that does not contain any solution found so far. If no

solution if found is a subinterval, this interval is not subdivided and the search procedure is stopped. That ensures the overall algorithm can not loop indefinitely. The corresponding **algorithm 7.1.1** presents this method by referring to the definitions defined above.

Algorithm 7.1.1 *function EPSDIC()*

Initialization

Determine the solution with the best cost $f_{start} = (x_{start}, \rho_{start})$ in a limited amount of time $TIME$.

Set ρ_{low} equal to 0 and ρ_{up} equal to ρ_{start}.

Set $Sol() = \{f_{start}\}$.

MAIN LOOP: $search(\rho_{low}, \rho_{up})$

Divide the exploration space $subint(\rho_{low}, \rho_{up})$ into two subintervals $upperint(\rho_{low}, \rho_{up})$ and $lowerrint(\rho_{low}, \rho_{up})$.

Solve problem $P_{lowerint(\rho_{low},\rho_{start})}$ within the time limit $TIME$:
if a solution y_{new} to this problem is found **then**
Add it to the list $Sol()$
Launch two other searches:
$search(\rho_{low}, \rho_{low} + \frac{1}{4} \times (\rho_{start} - \rho_{low}))$ and
$search(\rho_{low} + \frac{1}{4} \times (\rho_{start} - \rho_{low}), \rho_{low} + \frac{1}{2} \times (\rho_{start} - \rho_{low}))$.
end if

Solve problem $P_{upperint(\rho_{low},\rho_{start})}$ within the time limit $TIME$:
if a solution y_{new} to this problem is found **then**
Add it to the list $Sol()$
Launch two other searches:
$search(\rho_{low} + \frac{1}{2} \times (\rho_{start} - \rho_{low}), \rho_{low} + \frac{3}{4} \times (\rho_{start} - \rho_{low}))$ and
$search(\rho_{low} + \frac{3}{4} \times (\rho_{start} - \rho_{low}), \rho_{start}))$.
end if

return $Sol()$.

7.2 Greedy algorithm : *Greedy*

Among the different developed approaches, a greedy algorithm was developed to provide a good compromise between the effort to build the solutions and their qualities.

Notations:

- The set of available areas is denoted I, that of available sites J.
- N gives the number of sites connected to one area.
- Let consider the problem that minimizes the total opening cost such that the total capacity of the open sites is greater than the total demand. k_0 denotes the number of open sites in the solution of this problem.
- The solution of this problem gives a list of sites S, sorted in non-decreasing capacity order.
- Let A be the list of areas sorted according to one criterion, for instance they can be sorted in non-increasing demand order.
- Two other lists S^1 and S^2 are built from the list $(J \setminus S)$:
  - The first list, denoted S^1, consists of the set of sites whose capacity is bigger than that of the sites from S, sorted according to one criteria.
  - The second list, denoted S^2, is given by the set $(J \setminus \{S \cup S^1\})$, sorted according to one criteria.

The complete greedy **algorithm 7.2.1** is presented with two variations *Rate* (choose *RateOption* at the step **2. a)**) and *CostOption* (choose *Cost* at the step **2. a)**). Running totally the greedy algorithm with the variation *RateOption* provides a set of solutions that favors the rate ρ although running this algorithm with the variation *CostOption* provides a set that favors the cost.

Algorithm 7.2.1 *function greedy(A, S, S^1, S^2)*

for Iteration $k_0 \leq k \leq J$ **do**

1. Assign the first area $i = 1$ of A to the N last sites of S.

2. Iteratively, for each remaining area $i = 2, ..., N$ of A:

   for $n = 1, ..., N$

   a) **If** there is at least a site with enough remaining capacity to accept the additional demand $\frac{d_i}{N}$,

   RateOption Assign the area i to a site of S with enough left capacity to reach the lowest rate*

   CostOption Assign the area i to a site of S with enough left capacity to reach the lowest cost*

   b) **Otherwise**, remove the area i of S whose remaining capacity is the biggest to S^2 and substitute it by the first area of S^1 whose capacity is high enough, if it exists. Otherwise go to step 4.

   endfor

3. Save the solution, and calculate associated its cost and rate.

4. Afterwards:

   a) **If** $S^2 \neq \emptyset$, remove the first element of S^2, add it to S and update $k = |S| + 1$. Then, go to step 1.

   b) **Else If** $S^1 \neq \emptyset$, remove the first element of S^1, add it to S and update $k = |S| + 1$. Then, go to step 1.

   c) **Else** assign k to be equal to J+1 to end the building process.

end for

* *To get the lowest rate ρ (respectively cost), each feasible assignment is tested and the corresponding rate is assessed. Then, the assignment with the lowest rate (respectively cost) is chosen and perfomed.*

The function *greedy(A, S, S^1, S^2)* produces a list of solutions for a given problem by using four lists: the list of areas A and three lists of sites $-S, S^1$ and S^2- that are complementary and whose union gives the set I. For a given problem, the sites enumerated within these lists are defined in the **Notations**. It was chosen to sort each triplet of lists of sites S, S^1 and S^2 with the same criterion. However, the sorting criteria of these lists matter a lot, for two reasons:

- Sorting the areas in a different way may result in a different solution as each additional assignment is decided from the previous assignment decisions.
- Sorting the site in a different way may also result in a new solution as several assignment choices can result in the same objective function value. In this case, the first assignment decision that gives this best objective function value is chosen.

Each sorting criteria gives two symmetric lists, the one that follows a given sorting criterion in the increasing order and the one that follows that criterion in the opposite order. The criteria used to sort the list A and the triplet of lists (S, S^1, S^2) are given by the tables 7.1 and 7.2. The greedy algorithm can be runned by combining different couples A and (S, S^1, S^2) produced by using another couple of sorting criteria. Each combination provides often new solutions with the same greedy algorithm.

sorting criteria	value assigned to each site j	maximal number of new triplets
capacity level	Q_j	2×1
open cost	f_j	2×1
cumulative connection cost i $i = 1, ..., I$	$f_j + \sum_{k=1,...,i} c_{kj}$	$2 \times I$

Table 7.1: Sorting criteria for the triplet of lists (S, S^1, S^2).

sorting criteria	value assigned to each area i	maximal number of new triplets
demand	d_i	2×1
cumulative connection cost j $j = 1, ..., J$	$\sum_{k=1,...,j} c_{ik}$	$2 \times J$

Table 7.2: Sorting criteria for the list A.

7.3 The MMX model solved with Cplex: *MMX-Cplex*

Cplex can be used as an heuristic to provide the best found solution within a limited amount of time. Even if the best solution is not found, Cplex can provide at least the best feasible solution found so far.
The method Epsil is directly used and slightly adapted to our purpose.

After determining the solution with the best found cost $f_{start} = (x_{start}, y_{start})$ in a limited amount of time $TIME$, the exploration space is divided in P intervals where the first interval contains f_{start}. To finish, Cplex is used to minimize the cost by respecting the upper rate limit: the $p^{th}(p = P - 1$ to $1)$ interval corresponds to the problem $Epsil_{p,P}$:

$$\left\{ \begin{array}{ll} \min & Cost \\ s.t. & x \in X \\ & \rho \leq \frac{p}{P} \times y_{start} \end{array} \right\}$$

Remark: X is defined in the note `NT2/CRE_FT-LINA/v060529`.

The total exploration time is lower or equal to $TOTALTIME = P * TIME$. It can be noticed that the lower bound of the rate is unknown but 0 is a good approach

as it is common to reach a rate lower than 0.001 in real-sized problems. Moreover the more the upper rate limit decreases, the more the difficulty to solve such a problem increases.
For these reasons, it was preferred:

- to begin with the least constrained problem by going from the P-1th intervall to the first intervall
- to increase the amount of time per interval as soon as no solution is found

As the exploration is limited in time, the size of the remaining intervals is doubled and the number of intervals is divided by two to respect the initial time limit $TOTALTIME$, as described in the algorithm 7.3.1.

Algorithm 7.3.1 MMXCplex($TOTALTIME$,P)

Initialization:
Set $TIME$ equal to $\frac{TOTALTIME}{P}$.
Determining the solution with the best cost $f_{start} = (x_{start}, y_{start})$ in a limited amount of time $TIME$ via Cplex.

for $p = P - 1$ to 1 **do**
Solve the problem $Epsil_{p,P}$ within the time limit $TIME$
if Cplex provides no solution **then**
Set $TIME$ equal to $2 \times TIME$.
Set P equal to $\frac{P}{2}$.
Set p equal to $\frac{p}{2}$ if p is even, otherwise set p equal to $\frac{p-1}{2}$.
end if
end for

7.4 Computational results and determination of a mature building method

7.4.1 The method *EPS* and *EPSDIC*

The four figures 7.2, 7.3, 7.4 and 7.5 present a comparison between the exact solutions and the solutions given by solving the AVG model with the EPSIL method (called *EPS*) for the biggest available exactly-solved instances - `BiFLP12-5-1.VC.1`, `BiFLP12-5-2.VC.1`, `BiFLP12-5-3.VC.1` and `BiFLP17-5-1.VC.1`.

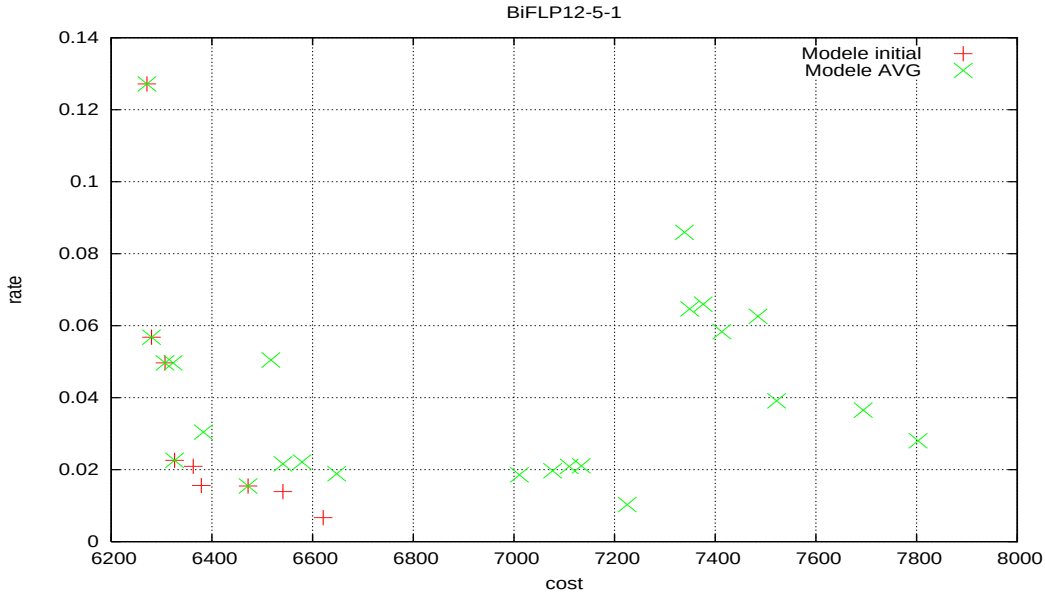


Figure 7.2: Exact and EPS solutions for a 12.5.1 instance

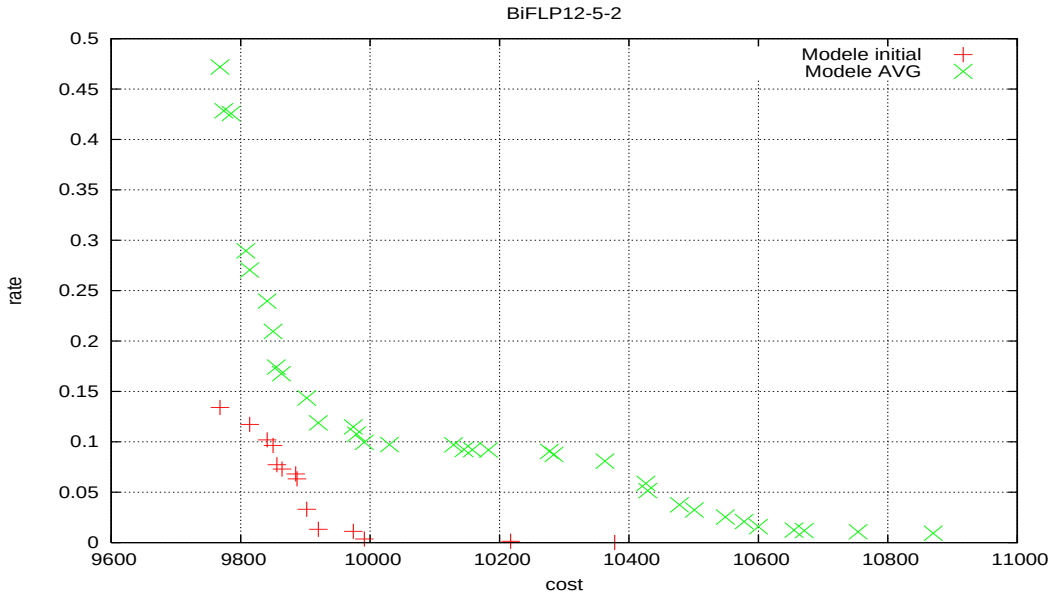


Figure 7.3: Exact and EPS solutions for a 12.5.2 instance

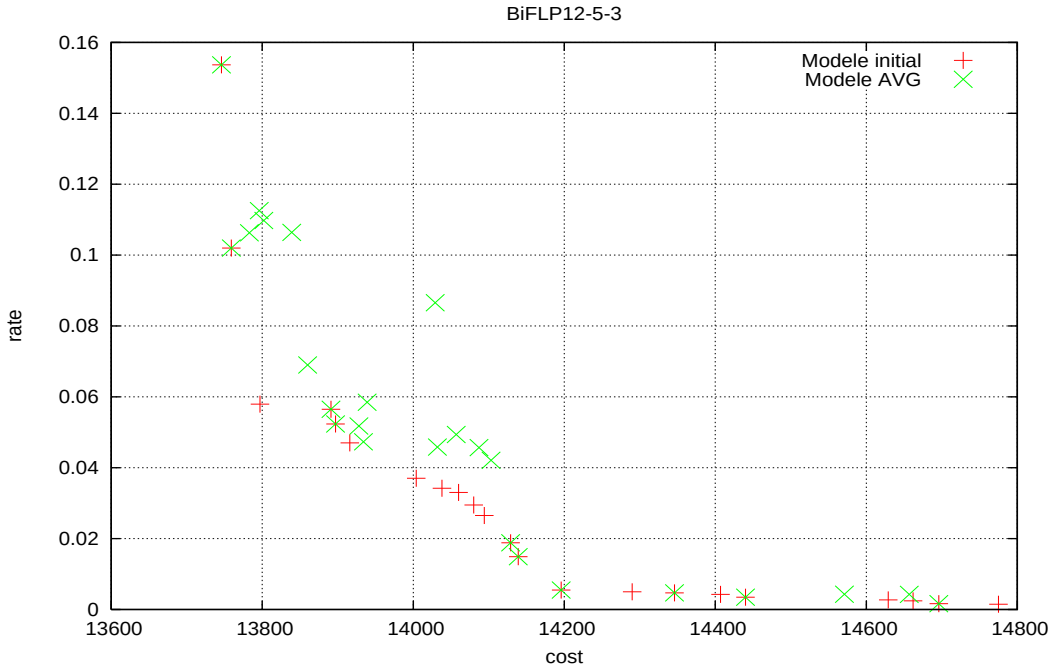


Figure 7.4: Exact and EPS solutions for a 12.5.3 instance

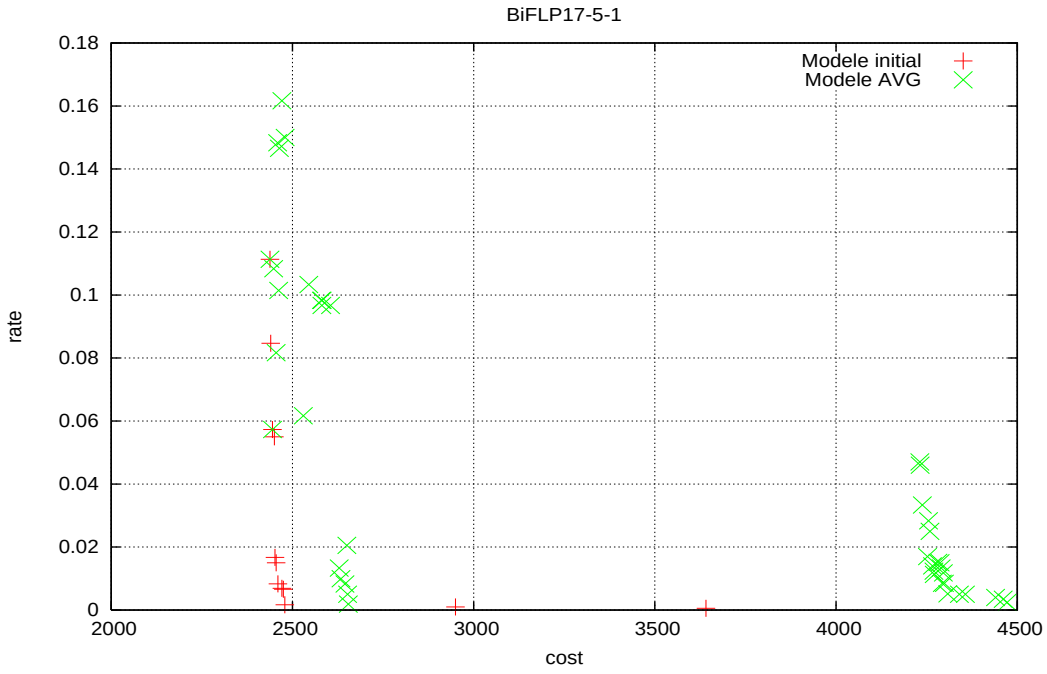


Figure 7.5: Exact and EPS solutions for a 17.5.1 instance

To compare the solution given by the methods *EPS* and *EPSDIC*, the four figures 7.6, 7.7, 7.8 and 7.9 present a comparison between the exact solutions and the solutions obtained by solving the AVG model with the EPSIL method with a dichotomous reduction of the search space (called *EPSDIC*) for the biggest exactly-solved instances - `BiFLP12-5-1.VC.1`, `BiFLP12-5-2.VC.1`, `BiFLP12-5-3.VC.1` and `BiFLP17-5-1.VC.1`.

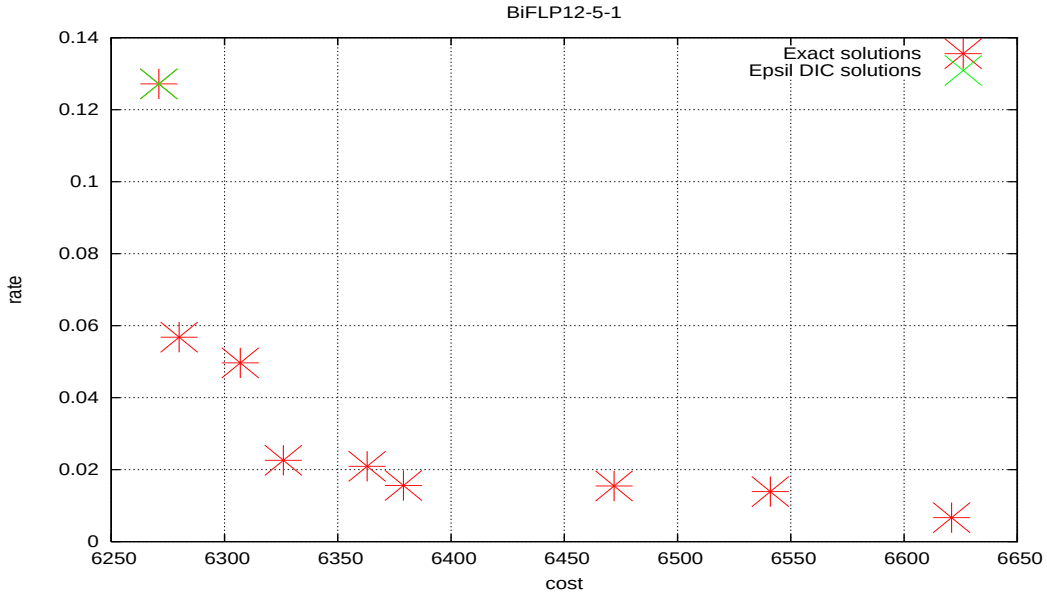


Figure 7.6: Exact and EPSDIC solutions for a 12.5.1 instance

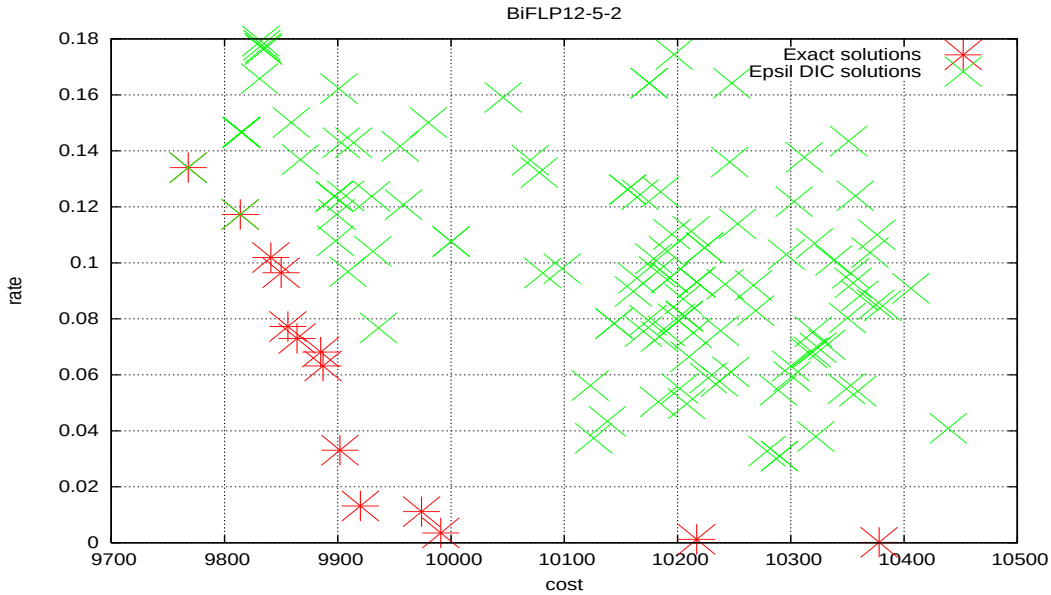


Figure 7.7: Exact and EPSDIC solutions for a 12.5.2 instance

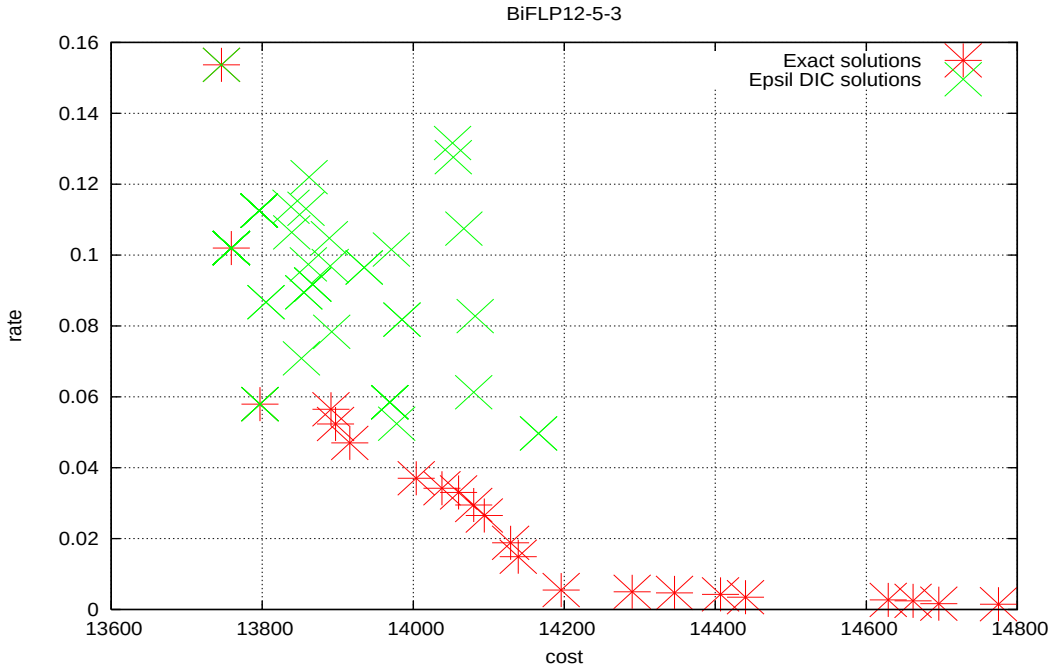


Figure 7.8: Exact and EPSDIC solutions for a 12.5.3 instance

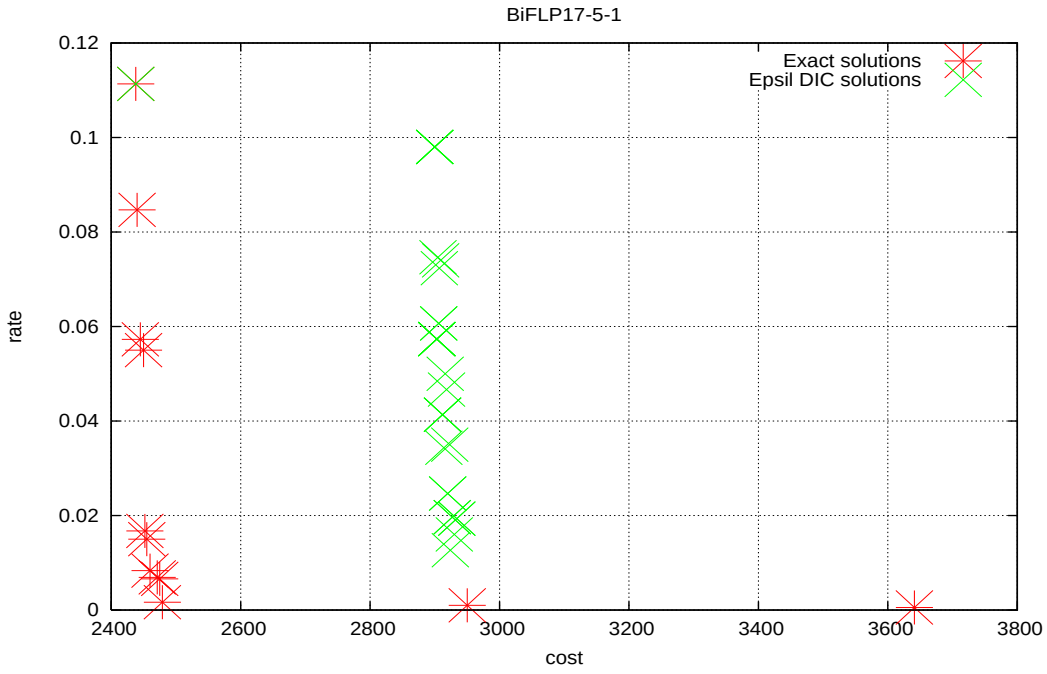


Figure 7.9: Exact and EPSDIC solutions for a 17.5.1 instance

The computational time and the number of solutions built by the algorithm *EPS* are summed up in the tables 7.3 and 7.4 for all the available instances. The computational time was limited to 1200 seconds. As shown by these tables, the number of solutions is small, especially for the problems that count 50 sites and areas, or at least 75 sites. These results showed that this method fails within a reasonable amount of time for medium-sized and big-sized problems.

instance	CPU_t (sec)	# solutions
BiFLP25-10.1.VC.1	1200	9
BiFLP25-10.1.VC.2	1200	7
BiFLP25-10.1.VC.3	1200	7
BiFLP25-10.1.VC.4	1200	6
BiFLP25-10.2.VC.1	670	19
BiFLP25-10.2.VC.2	890	15
BiFLP25-10.2.VC.3	986	14
BiFLP25-10.2.VC.4	986	15
BiFLP25-10.3.VC.1	393	38
BiFLP25-10.3.VC.2	740	21
BiFLP25-10.3.VC.3	982	32
BiFLP25-10.3.VC.4	893	41
BiFLP50-25.1.VC.1	1200	4
BiFLP50-25.1.VC.2	1200	3
BiFLP50-25.1.VC.3	1188	5
BiFLP50-25.1.VC.4	1200	4
BiFLP50-25.2.VC.1	1200	26
BiFLP50-25.2.VC.2	1200	22
BiFLP50-25.2.VC.3	1200	23
BiFLP50-25.2.VC.4	1200	17
BiFLP50-25.3.VC.1	1200	16
BiFLP50-25.3.VC.2	1200	14
BiFLP50-25.3.VC.3	1200	15
BiFLP50-25.3.VC.4	1200	18
BiFLP50-50.1.VC.x *	1200	$1 \leq ... \leq 2$
BiFLP50-50.2.VC.x *	1200	$1 \leq ... \leq 3$
BiFLP50-50.3.VC.x *	1200	$1 \leq ... \leq 3$

* x $\in \{1, 2, 3, 4\}$

Table 7.3: Computational results for the algorithm *EPS* for the instances `BiFLP25-10-x.VC.x`, `BiFLP50-25-x.VC.x` and `BiFLP50-50-x.VC.x` .

instance	CPU_t (sec)	# solutions
BiFLP75-25.1.VC.1	1200	4
BiFLP75-25.1.VC.2	1200	2
BiFLP75-25.1.VC.3	1200	3
BiFLP75-25.1.VC.4	1200	4
BiFLP75-25.2.VC.1	1200	5
BiFLP75-25.2.VC.2	1200	3
BiFLP75-25.2.VC.3	1200	4
BiFLP75-25.2.VC.4	1200	5
BiFLP75-25.3.VC.1	1200	3
BiFLP75-25.3.VC.2	1200	5
BiFLP75-25.3.VC.3	1200	3
BiFLP75-25.3.VC.4	1200	5
BiFLP75-50.1.VC.1	1200	5
BiFLP75-50.1.VC.2	1200	3
BiFLP75-50.1.VC.3	1200	3
BiFLP75-50.1.VC.4	1200	2
BiFLP75-50.2.VC.1	1200	2
BiFLP75-50.2.VC.2	1200	3
BiFLP75-50.2.VC.3	1200	2
BiFLP75-50.2.VC.4	1200	4
BiFLP75-50.3.VC.1	1200	4
BiFLP75-50.3.VC.2	1200	3
BiFLP75-50.3.VC.3	1200	2
BiFLP75-50.3.VC.4	1200	3
BiFLP75-75.1.VC.1	1200	1
BiFLP75-75.1.VC.2	1200	1
BiFLP75-75.1.VC.3	1200	1
BiFLP75-75.1.VC.4	1200	2
BiFLP100-30.N.VC.x *	1200	1
BiFLP100-60.N.VC.x *	1200	1
BiFLP100-80.N.VC.x *	1200	1
BiFLP100-100.N.VC.x *	1200	1

* x $\in \{1, 2, 3, 4\}, N \in \{1, 2, 3\}$

Table 7.4: Computational results for the algorithm *EPS* for the instances `BiFLP75-25-x.VC.x`, `BiFLP75-50-x.VC.x` and some other bigger instances .

The computational time and the number of solutions built by the algorithm *EPSDIC* are summed up in the tables 7.5 and 7.6 for all the available instances. The computational time was limited to 1200 seconds. As shown by these tables, the number of solutions is small, especially for the problems that count 50 sites and areas, or at least 75 sites. Similarly to the method *EPS*, these results showed that this method fails within a reasonable amount of time for medium-sized and big-sized problems.

instance	CPU_t (sec)	# solutions
BiFLP25-10.1.VC.1	20	2
BiFLP25-10.1.VC.2	451	68
BiFLP25-10.1.VC.3	1201	17
BiFLP25-10.1.VC.4	1200	16
BiFLP25-10.2.VC.1	1200	18
BiFLP25-10.2.VC.2	101	193
BiFLP25-10.2.VC.3	986	145
BiFLP25-10.2.VC.4	986	145
BiFLP25-10.3.VC.1	1200	32
BiFLP25-10.3.VC.2	740	213
BiFLP25-10.3.VC.3	1200	45
BiFLP25-10.3.VC.4	1200	45
BiFLP50-25.1.VC.1	1200	4
BiFLP50-25.1.VC.2	1200	2
BiFLP50-25.1.VC.3	1188	3
BiFLP50-25.1.VC.4	1200	2
BiFLP50-25.2.VC.1	1200	7
BiFLP50-25.2.VC.2	1200	4
BiFLP50-25.2.VC.3	1200	8
BiFLP50-25.2.VC.4	1200	4
BiFLP50-25.3.VC.1	1200	18
BiFLP50-25.3.VC.2	1200	25
BiFLP50-25.3.VC.3	1200	2
BiFLP50-25.3.VC.4	1200	14
BiFLP50-50.1.VC.x *	1200	1
BiFLP50-50.2.VC.x *	1200	1
BiFLP50-50.3.VC.x *	1200	1

* x $\in \{1, 2, 3, 4\}$

Table 7.5: Computational results for the algorithm *EPSDIC* for the instances `BiFLP25-10-x.VC.x`, `BiFLP50-25-x.VC.x` and `BiFLP50-50-x.VC.x` .

instance	CPU_t (sec)	# solutions
BiFLP75-25.1.VC.1	1200	2
BiFLP75-25.1.VC.2	1200	1
BiFLP75-25.1.VC.3	1200	2
BiFLP75-25.1.VC.4	1200	2
BiFLP75-25.2.VC.1	1200	2
BiFLP75-25.2.VC.2	1204	2
BiFLP75-25.2.VC.3	1205	3
BiFLP75-25.2.VC.4	1201	4
BiFLP75-25.3.VC.1	1202	4
BiFLP75-25.3.VC.2	1200	6
BiFLP75-25.3.VC.3	1207	2
BiFLP75-25.3.VC.4	1200	6
BiFLP75-50.1.VC.1	1202	7
BiFLP75-50.1.VC.2	1200	1
BiFLP75-50.1.VC.3	1200	1
BiFLP75-50.1.VC.4	1200	1
BiFLP75-50.2.VC.1	1200	2
BiFLP75-50.2.VC.2	1200	1
BiFLP75-50.2.VC.3	1200	1
BiFLP75-50.2.VC.4	1200	1
BiFLP75-50.3.VC.1	1200	2
BiFLP75-50.3.VC.2	1202	2
BiFLP75-50.3.VC.3	1200	1
BiFLP75-50.3.VC.4	1200	1
BiFLP75-75.1.VC.1	1200	1
BiFLP75-75.1.VC.2	1200	1
BiFLP75-75.1.VC.3	1200	1
BiFLP75-75.1.VC.4	1200	1
BiFLP100-30.N.VC.x *	1200	1
BiFLP100-60.N.VC.x *	1200	1
BiFLP100-80.N.VC.x *	1200	1
BiFLP100-100.N.VC.x *	1200	1

* x $\in \{1, 2, 3, 4\}, N \in \{1, 2, 3\}$

Table 7.6: Computational results for the algorithm *EPSDIC* for the instances `BiFLP25-10-x.VC.x`, `BiFLP50-25-x.VC.x` and `BiFLP50-50-x.VC.x` .

7.4.2 The method *Greedy*

Note that the exact solutions for instances bigger than that presented in this subsection can not be solved. The four figures 7.10, 7.11, 7.12 and 7.13 present a comparison between the exact solutions and the greedy solutions for the biggest exactly-solved instances - `BiFLP12-5-1.VC.1`, `BiFLP12-5-2.VC.1`, `BiFLP12-5-3.VC.1` and `BiFLP17-5-1.VC.1`.

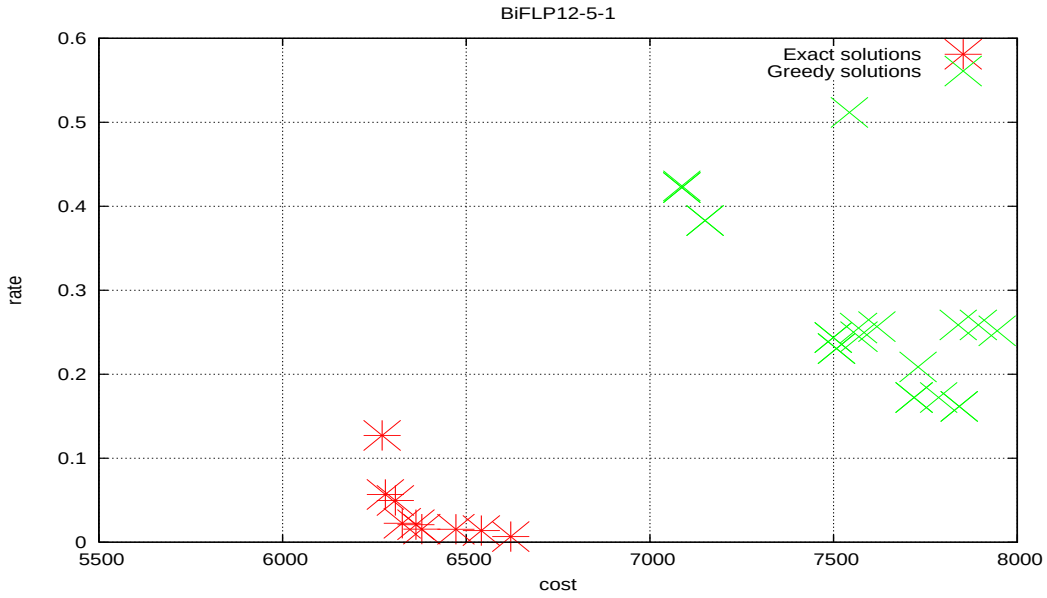


Figure 7.10: Exact and greedy solutions for a 12.5.1 instance

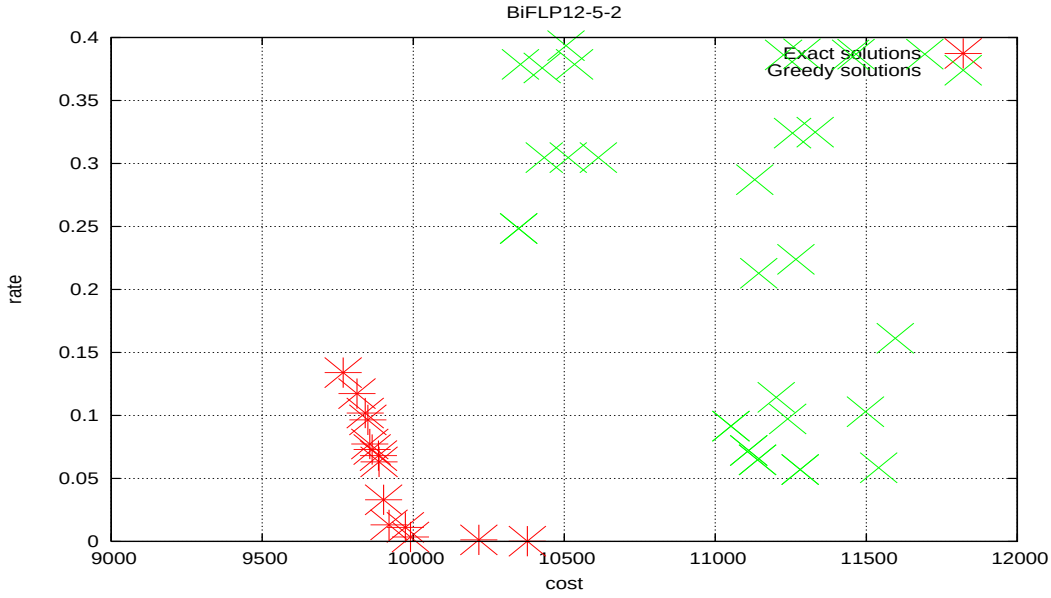


Figure 7.11: Exact and greedy solutions for a 12.5.2 instance

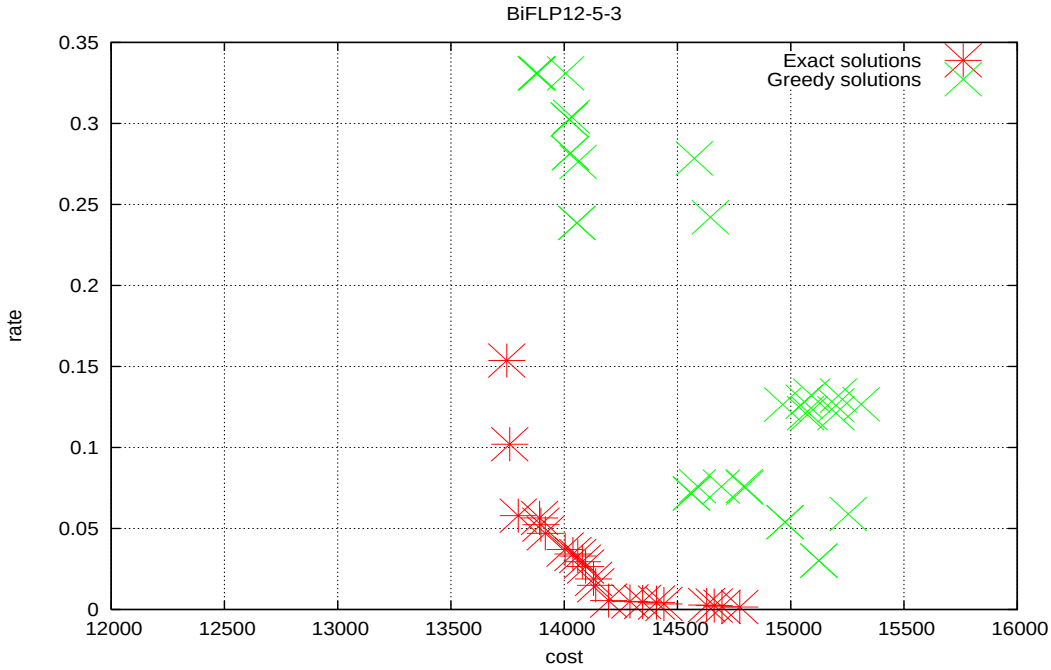


Figure 7.12: Exact and greedy solutions for a 12.5.3 instance

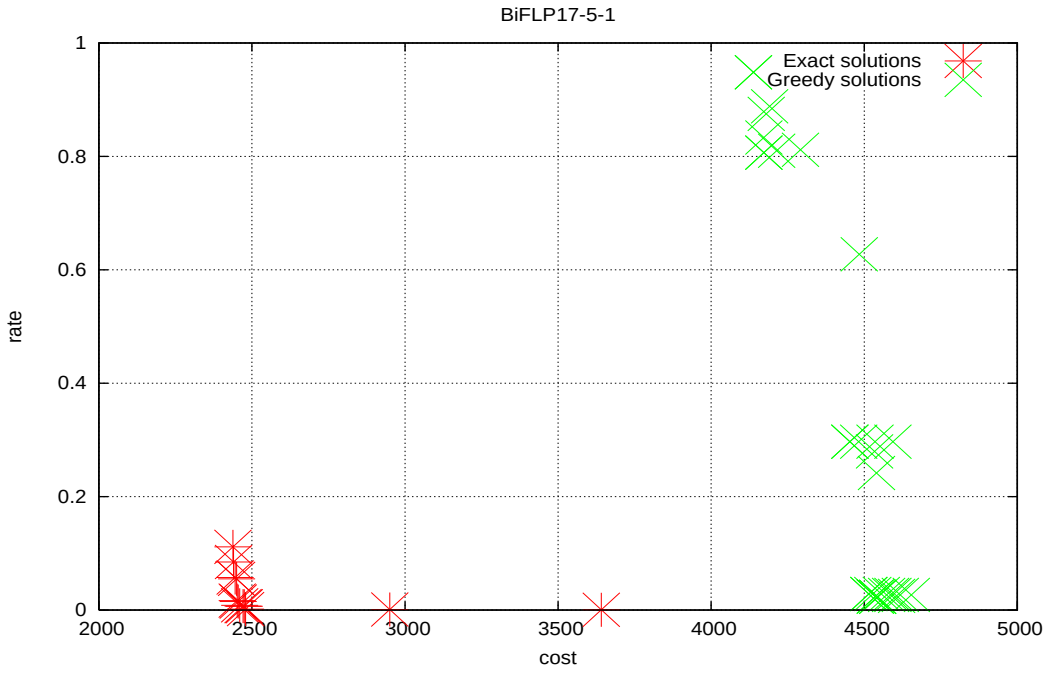


Figure 7.13: Exact and greedy solutions for a 17.5.1 instance

The solution provided by the greedy algorithm can be fairly splitted into two groups: those with a relatively low rate ρ and a high cost and those with a relatively high rate ρ and a low cost. These two groups appear more obvious with medium and big-sized instances. To illustrate these characteristics, the three graphes 7.14, 7.15 and 7.16 present the solutions built by this algorithm.

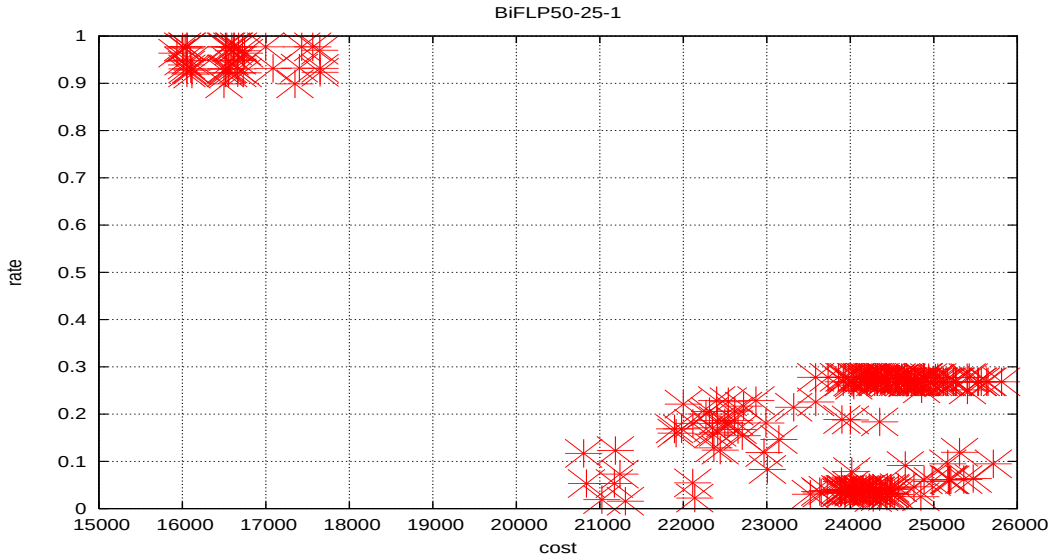


Figure 7.14: Greedy solutions with a 50-25-1 instance

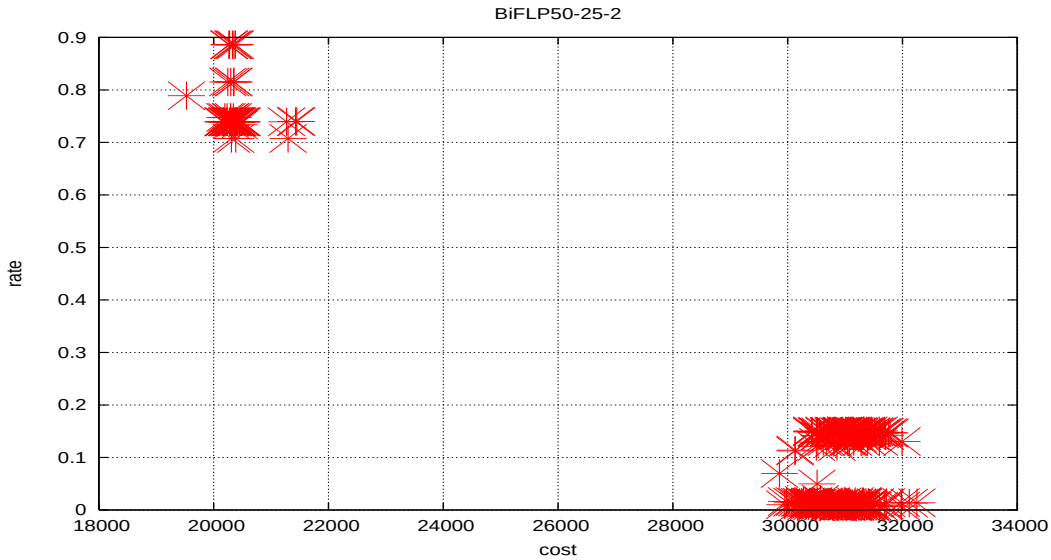


Figure 7.15: Greedy solutions with a 50-25-2 instance

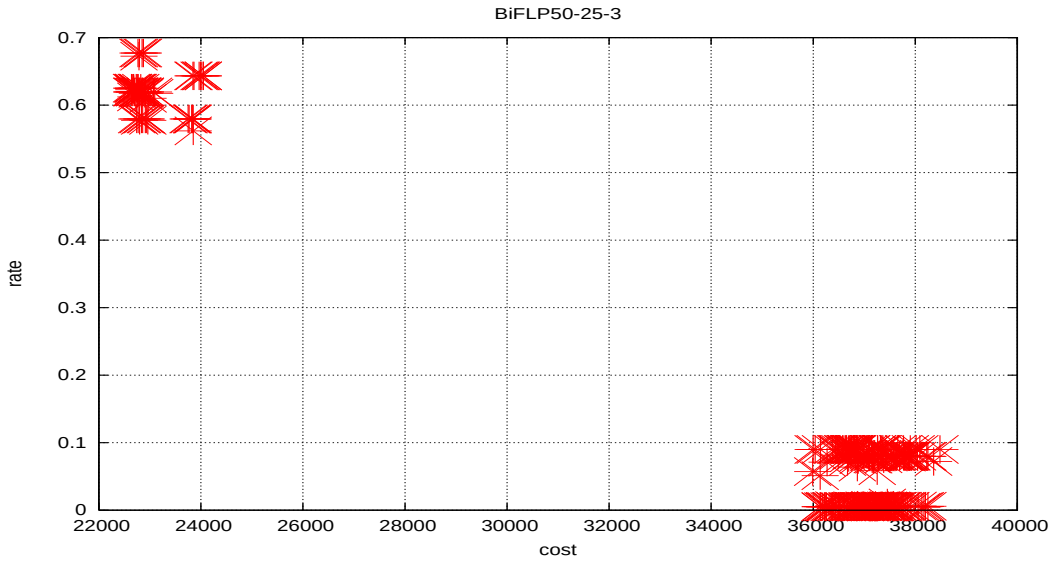


Figure 7.16: Greedy solutions with a 50-25-3 instance

The computational time and the number of solutions built by this algorithm are summed up in the tables 7.7, 7.8 and 7.9 for all the available instances. As show by these tables, the computational time lies between one second and 2623 seconds. The number of non-identical solutions is relatively constant, between 60 and 83. The number of non-dominated solutions lies between 3 and 13.

instance	CPU_t (sec)	# solutions	# non-dominated solutions
BiFLP25-10.1.VC.1	0	73	6
BiFLP25-10.1.VC.2	0	65	10
BiFLP25-10.1.VC.3	1	60	8
BiFLP25-10.1.VC.4	0	60	8
BiFLP25-10.2.VC.1	1	73	10
BiFLP25-10.2.VC.2	0	79	9
BiFLP25-10.2.VC.3	1	75	5
BiFLP25-10.2.VC.4	0	75	5
BiFLP25-10.3.VC.1	1	80	10
BiFLP25-10.3.VC.2	1	79	9
BiFLP25-10.3.VC.3	1	80	10
BiFLP25-10.3.VC.4	0	80	10
BiFLP50-25.1.VC.1	11	77	7
BiFLP50-25.1.VC.2	11	83	13
BiFLP50-25.1.VC.3	10	78	8
BiFLP50-25.1.VC.4	11	80	10
BiFLP50-25.2.VC.1	20	78	8
BiFLP50-25.2.VC.2	21	81	11
BiFLP50-25.2.VC.3	20	80	10
BiFLP50-25.2.VC.4	21	79	9
BiFLP50-25.3.VC.1	29	79	9
BiFLP50-25.3.VC.2	30	80	10
BiFLP50-25.3.VC.3	30	79	9
BiFLP50-25.3.VC.4	30	78	7
BiFLP50-50.1.VC.1	57	78	8
BiFLP50-50.1.VC.2	56	78	8
BiFLP50-50.1.VC.3	56	74	4
BiFLP50-50.1.VC.4	56	77	7
BiFLP50-50.2.VC.1	110	80	10
BiFLP50-50.2.VC.2	111	78	8
BiFLP50-50.2.VC.3	111	83	13
BiFLP50-50.2.VC.4	110	79	9
BiFLP50-50.3.VC.1	164	77	7
BiFLP50-50.3.VC.2	164	78	8
BiFLP50-50.3.VC.3	163	80	10
BiFLP50-50.3.VC.4	164	83	13

Table 7.7: Computational results for the greedy algorithm for the instances `BiFLP25-10-x.VC.x`, `BiFLP50-25-x.VC.x` and `BiFLP50-50-x.VC.x` .

instance	CPU_t (sec)	# solutions	# non-dominated solutions
BiFLP75-25.1.VC.1	20	76	6
BiFLP75-25.1.VC.2	20	77	7
BiFLP75-25.1.VC.3	21	76	6
BiFLP75-25.1.VC.4	20	78	8
BiFLP75-25.2.VC.1	40	74	4
BiFLP75-25.2.VC.2	39	78	8
BiFLP75-25.2.VC.3	39	80	10
BiFLP75-25.2.VC.4	40	75	5
BiFLP75-25.3.VC.1	57	78	8
BiFLP75-25.3.VC.2	57	77	7
BiFLP75-25.3.VC.3	58	77	7
BiFLP75-25.3.VC.4	58	76	6
BiFLP75-50.1.VC.1	18	16	3
BiFLP75-50.1.VC.2	102	76	6
BiFLP75-50.1.VC.3	102	76	6
BiFLP75-50.1.VC.4	101	73	3
BiFLP75-50.2.VC.1	202	75	5
BiFLP75-50.2.VC.2	202	75	5
BiFLP75-50.2.VC.3	201	77	7
BiFLP75-50.2.VC.4	201	78	8
BiFLP75-50.3.VC.1	299	78	8
BiFLP75-50.3.VC.2	299	75	5
BiFLP75-50.3.VC.3	299	76	6
BiFLP75-50.3.VC.4	300	76	6
BiFLP75-75.1.VC.1	252	75	5
BiFLP75-75.1.VC.2	260	73	3
BiFLP75-75.1.VC.3	255	75	5
BiFLP75-75.1.VC.4	255	73	3
BiFLP75-75.2.VC.1	548	74	4
BiFLP75-75.2.VC.2	554	74	4
BiFLP75-75.2.VC.3	552	76	6
BiFLP75-75.2.VC.4	553	77	7
BiFLP75-75.3.VC.1	825	75	4
BiFLP75-75.3.VC.2	831	77	7
BiFLP75-75.3.VC.3	829	75	5
BiFLP75-75.3.VC.4	828	75	5

Table 7.8: Computational results for the greedy algorithm for the instances `BiFLP75-x-x.VC.x` .

instance	CPU_t (sec)	# solutions	# non-dominated solutions
BiFLP100-30.1.VC.1	50	75	5
BiFLP100-30.1.VC.2	51	75	5
BiFLP100-30.1.VC.3	49	76	6
BiFLP100-30.1.VC.4	49	77	7
BiFLP100-30.2.VC.1	48	76	6
BiFLP100-30.2.VC.2	95	76	6
BiFLP100-30.2.VC.3	96	74	4
BiFLP100-30.2.VC.4	97	78	8
BiFLP100-30.3.VC.1	143	76	6
BiFLP100-30.3.VC.2	141	76	6
BiFLP100-30.3.VC.3	141	78	8
BiFLP100-30.3.VC.4	142	77	7
BiFLP100-60.1.VC.1	248	75	5
BiFLP100-60.1.VC.2	247	74	4
BiFLP100-60.1.VC.3	248	77	7
BiFLP100-60.1.VC.4	248	75	5
BiFLP100-60.2.VC.1	490	76	6
BiFLP100-60.2.VC.2	491	76	6
BiFLP100-60.2.VC.3	490	77	7
BiFLP100-60.2.VC.4	490	76	6
BiFLP100-60.3.VC.1	729	76	6
BiFLP100-60.3.VC.2	727	76	6
BiFLP100-60.3.VC.3	729	77	7
BiFLP100-60.3.VC.4	729	77	7
BiFLP100-80.1.VC.1	498	75	5
BiFLP100-80.1.VC.2	500	76	6
BiFLP100-80.1.VC.3	498	74	4
BiFLP100-80.1.VC.4	499	74	4
BiFLP100-80.2.VC.1	1000	75	5
BiFLP100-80.2.VC.2	1753	77	7
BiFLP100-80.2.VC.3	999	74	4
BiFLP100-80.2.VC.4	1003	74	4
BiFLP100-80.3.VC.1	2606	78	8
BiFLP100-80.3.VC.2	1493	75	5
BiFLP100-80.3.VC.3	1489	76	6
BiFLP100-80.3.VC.4	1489	76	6
BiFLP100-100.1.VC.1	858	75	5
BiFLP100-100.1.VC.2	860	74	4
BiFLP100-100.1.VC.3	288	16	3
BiFLP100-100.1.VC.4	860	75	5
BiFLP100-100.2.VC.1	1753	75	5
BiFLP100-100.2.VC.2	1752	75	5
BiFLP100-100.2.VC.3	1752	75	5
BiFLP100-100.2.VC.4	1747	77	7
BiFLP100-100.3.VC.1	2623	76	6
BiFLP100-100.3.VC.2	2621	75	5
BiFLP100-100.3.VC.3	2619	78	8
BiFLP100-100.3.VC.4	2618	76	6

Table 7.9: Computational results for the greedy algorithm for the instances `BiFLP100-x-x.VC.x` .

7.4.3 The method *MMXCplex*

This method is more efficient than the method *Greedy* as most of the solutions provided by the algorithm *MMXCplex* dominate those provided by the greedy algorithm. However the greedy algorithm provides most of the time some solutions with a lower rate ρ than that of the solutions built by the algorithm *MMXCplex*.
To illustrate this behavior, an exemple was plotted for the instance `BiFLP100-100-3.VC.1`. The parameters used with the method *MMXCplex* were: P=30 and $TOTALTIME$= 2000 seconds. In the figure 7.17, it can be noticed in the right-lower corner that the rate ρ of three greedy solutions is lower than that of all the other MMXCplex solutions.

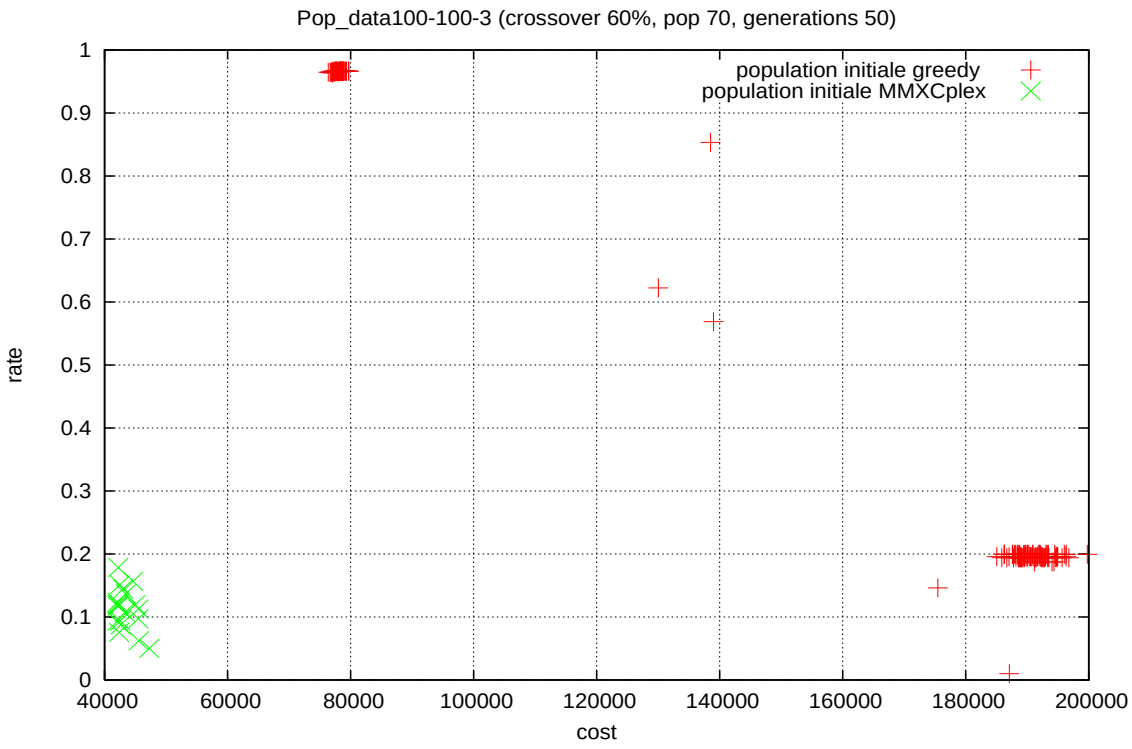


Figure 7.17: Greedy and MMCplex solutions for a 100.100.3 instance

As a consequence, both methods are complementary as the method *Greedy* provides solutions with a relatively better rate ρ in most cases (by setting the parameter $TOTALTIME$ equal to the time spent by the greedy algorithm) and the method *MMXCplex* builds solutions with a relatively better cost.
Experimental tests on the available instances have proved that these solutions with a relatively good rate ρ are obtained by using the ordering criteria of the table 7.10. Up to four different solutions with a low rate ρ can be obtained. The computational time to build these solutions increases up to 62 seconds for the largest solutions.

Table 7.10: Selected sorting criteria for the triplet of lists (A, S, S^1, S^2) used by the function $greedy(A, S, S^1, S^2)$.

Criteria ID	assigned value	ordering criterion
Rule # 1	site $j \leftarrow Q_j$	increasing
	area $i \leftarrow d_i$	decreasing
Rule # 2	site $j \leftarrow Q_j$	increasing
	area $i \leftarrow \sum_{k=1,\ldots,j} c_{ik},\ j = 1$	increasing
Rule # 3	site $j \leftarrow f_j + \sum_{k=1,\ldots,i} c_{kj},\ i = 1$	increasing
	area $i \leftarrow d_i$	decreasing
Rule # 4	site $j \leftarrow f_j$	increasing
	area $i \leftarrow d_i$	decreasing

The results of this section allows to finalize a building **algorithm 7.4.1** to create the initial population used by the MOEA. The function $bestRate(P)$ returns the lowest value of the rate ρ among the solutions within the list P.

Algorithm 7.4.1 function $Build(TOTALTIME, P)$

Initialization
Set $P \leftarrow \emptyset$
Set $P_{temp} \leftarrow \emptyset$

Main loop
Set $P \leftarrow P \cup \{MMXCplex(TOTALTIME, P)\}$
$\rho_{inf} \leftarrow bestRate(P)$
for $1 \leq i \leq 4$ **do**
Sort the triplet of lists (A, S, S^1, S^2) according to the criterion *Rule # i*
Set $P_{temp} \leftarrow P_{temp} \cup greedy(A, S, S^1, S^2)$
end for
$\rho_{greedy_inf} \leftarrow bestRate(P_{temp})$
if $\rho_{greedy_inf} < \rho_{inf}$ **then**
Set $P \leftarrow P \cup P_{temp}$
end if

Return P

The results of this subsection *MMXCplex* permits to rule out the methods *EPS* and *EPSDIC* due to the unability of these methods to generate several solutions within a reasonable amount of time. Finally it is decided to use together the methods *MMXCplex* and *Greedy* via the function $Build(TOTALTIME, P)$.

Chapter 8

Multiple-objective evolutionary algorithm (MOEA) to solve the bi-objective location problem

8.1 Presentation of SPEA

8.1.1 Strengths and weaknesses

SPEA (Strenght Pareto Evolutionnary Algorithm) is a multi-objective evolutionary metaheuristic (MOEA) based on a Pareto dominance-based evaluation. This algorithm was developed to solve problems with continuous variables, as well as problems with non-linear variables or discontinuous feasible space.

This algorithm was successfully applied on many problems such as:
- the multi-dimensional knapsack problem [ZT99]
- the set packing problem [DGD05]

This algorithm is considered to be efficient for providing a good estimation of the intermediate solutions of the set of the non-dominated solutions. However the drawback of this algorithm is its lack of efficiency for providing a satisfying estimation of the extreme solutions, i.e. those which minimize one of the objective functions.

8.1.2 General structure

SPEA uses a population, an archive (external set) and a mating pool.

All the current approximation of the non-dominated solutions form the archive. The regular population is defined as the set of feasible solutions that are dominated. The archive is used to prevent from loosing the non-dominated solutions found through all the evolutionary process. The fitness value assigned to each solution of the population and the archive is necessary for comparing the solutions between them with respect to several objective functions. The mating pool is a restrited union of the regular population and the archive: the evolutionary operators are applied only on the mating pool members.

Each iteration, also called generation, of this algorithm consists of:
- updating the archive from the current non-dominated population members,

- assessing a fitness value to both archive and population members,
- performing a binary tournament of the set given by the union of the archive and the population (by picking iteratively two random members of this set and selecting the fittest one) to create a mating pool,
- performing recombination and mutation,
- and replacing the old population by the resulting offspring population (i.e. the next population).

Among the characteristics of this algorithm, there are :

- The fitness of the solutions depends on their Pareto-dominance ranking.
- The storage of the non-dominated solutions is done through the evolutionary process in the archive.
- If the size of the updated archive exceeds a predefined limit, further archive members are deleted by a clustering technique which preserves the characteristics of the nondominated front.
- Any dominated individuals or duplicates (regarding the objective values) are removed from the archive during its update.

8.2 Aggressive-SPEA

The Aggressive SPEA (A-SPEA) algorithm [DGD05] is an enhancement of SPEA that was developed to make the SPEA more aggressive by collecting all non-dominated solutions and by introducing a neighborhood search operator (path-relinking). Before going further, several notations have to be defined.This algorithm is modified to satisfy the characteristics of the problem defined by the model MMX: the fitness value procedure remains unchanged and the selection procedure is slightly changed.

Notations:

- the regular population P
- the size of the regular population is $populationSize$
- the set of current non-dominated solutions $\overline{P}$
- the mating pool P'
- the next population P''

Among the components of A-SPEA, the fitness assessment of the solutions and the procedure selection have been employed in our MOEA with a slight modification of the selection procedure.

Fitness value

Before assessing the fitness value of each solution, the set of potential non-dominated solutions (denoted by $\overline{P}$) has to be updated at the beginning of each loop. Once it has been done, the fitness value of these solutions is assessed before that of the dominated solutions as follows:

- the fitness value $Fitness_s$ of a non-dominated solution s is given by the number of solutions that dominate it divided by the number of non-dominated solutions plus one,

$$Fitness_s = \frac{\{|s' \in P, s \geq s'|\}}{1+|P|}, \forall s \in \overline{P}$$

- the fitness value $Fitness_s$ of a dominated solution s is determined by the sum of the fitness value of the non-dominated solutions that dominates if plus one.

$$Fitness_s = 1 + \sum_{s' \in \overline{P}, s' \geq s} Fitness_s, \forall s \in P$$

It can be noticed that the fitness value is always less than one for non-dominated solutions and strictly greater than one for dominated solutions.

Selection
After updating the archive set $\overline{P}$, all the solutions belonging to this set are excluded from the current population P. As the mating pool P' is built by taking solutions from the sets $\overline{P}$ and P, it is important to prevent any solution from belonging to these two set simultaneously and from biaising the fitness value of the solutions.
Then, the selection procedure (see the **algorithm 8.2.1**) builds the mating pool: two individuals are randomly selected from the set $\overline{P} \cup P$ and the individual with the lowest fitness value wins the binary tournament and is added to the mating pool.

However, we consider that two solutions are equal only and only if their decisions variables are equal. Therefore, two solutions whose objective values are equal but with not exactly the same decision variables are not considered equal. An important case to consider is two solutions –s_a and s_{2b},where $s_a \neq s_2$– but with the same fitness value –$Fitness_{s_a} = Fitness_{s_b}$.
The existence of these solutions may lead to two scenarios:

- If s_a and s_b compete in the same tournament, the winner is selected randomly exactly as the A-SPEA did.

- If one of these two solutions, say s_a, was already put in the mating pool, the other one s_b can be integrated in the mating pool if this solution wins a binary tournament although the A-SPEA selection procedure would reject this solution s_a out of the mating pool.

Algorithm 8.2.1 *function Popselection(PopulationSize, $P, \overline{P}$)*

Fitness evaluation

$Fitness_s \leftarrow \frac{\{|s' \in P, s \geq s'|\}}{1+|P|}, \forall s \in \overline{P}$

$Fitness_s \leftarrow 1 + \sum_{s' \in \overline{P}, s' \geq s} Fitness_s, \forall s \in P$

Popselection

while $|P'| < min(PopulationSize, P, \overline{P})$ **do**

$(s_1, s_2) \leftarrow RandomSelect(s \in P \cup \overline{P} \setminus P')$

if $Fitness_{s_1} < Fitness_{s_2}$

$P' \leftarrow P' \cup s_1$

else

$P' \leftarrow P' \cup s_2$

endIf

end while

return P'

8.3 The final MOEA: A-SPEA2

8.3.1 Presentation

Our MOEA is inspired by the A-SPEA algorithm. Concerning the bi-objective location problem, our MOEA manages the population at each generation as follows:

- A mating pool P' is built by selecting some of the archive members and population members given after a binary tournament. The size of this mating pool equals to the size of the population *Populationsize*.
- Then the crossover operator is applied to the members of the mating pool, between one solution from the archive $\overline{P}$ and one solution from the population P.
  - A crossover operator gives two offpsring.
  - Each unfeasible offspring is repaired twice, by a cost-friendly building algorithm and a rate-friendly building algorithm, and up to two feasible solutions can be obtained from each unfeasible offspring.
  - All the feasible offspring are added to the next population P'' (i.e. up to 4 offspring from a crossover of 2 parents).
- After performing crossover, all the solutions of population P' and the archive $\overline{P}$ are copied to the next population P''.
- Then a mutation operator is applied to the current population P'' as a tool to increase the diversity of the next population before reducing it:
  - The mutation operator gives one offpsring.
  - If unfeasible, the offspring is repaired twice, by a cost-friendly building algorithm and a rate-friendly building algorithm, and up to two feasible solutions can be obtained.
  - All the feasible offspring are added to the next population P'' (i.e. up to 2 offspring from one mutation of a parent).

- Then the archive $\overline{P}$ is updated:
  - by moving all new non-dominated solutions from P'' and adding them to the archive
  - by removing in the archive all dominated solutions.
- The path-relinking is applied to the archive
  - The solutions given by the path-relinking process are put into the next populationP''
  - At the end of this process, the archive $\overline{P}$ is again updated.

- The population P is built by picking all the remaining solutions from P'' and reducing to the desired size $Populationsize$ (see the **algorithm 8.3.1**):

-step1 the set of non-dominated solutions from P'' are moved to the population P if the size of this set does not lead to exceed the desired size $Populationsize$

-step2 otherwise solutions are randomly selected among this set and moved from P'' to P

-step3 if the size of the population P is still lower than $Populationsize$, repeat the step [*step1*]

Algorithm 8.3.1 *function ReducePop(P,Populationsize)*

$P_{reduced} \leftarrow \emptyset$
$P_{buffer} \leftarrow \emptyset$

while $|P_{reduced}| < Populationsize$ **do**
  $P_{buffer} \leftarrow \{s \in P, \nexists s' \in P, s' \succ s\}$
  if $|P_{buffer}| + |P_{reduced}| < Populationsize$ **then**
    $P \leftarrow \{P \setminus P_{buffer}\}$
    $P_{reduced} \leftarrow \{P_{reduced} \cup P_{buffer}\}$
    $P_{buffer} \leftarrow \emptyset$
  end if
  else
    while $s \leftarrow RandomSelect(P_{buffer})$ **do**
      $s \leftarrow RandomSelect(P_{buffer})$
      $P_{reduced} \leftarrow \{P_{reduced} \cup \{s\}\}$
    end while
  end else
end while

return $P_{reduced}$.

8.3.2 Algorithm

The algorithm of the MOEA developed for the bi-objective location problem is described below (see **algorithm 8.3.2**):

Algorithm 8.3.2 *function A-SPEA2(PopulationSize, CrossoverProb, MutationProb, StoppingCondition, TOTALTIME, P)*

Initialization

$\overline{P} \leftarrow Build(TOTALTIME, P)$
$\overline{P} \leftarrow \emptyset$
$\overline{P} \leftarrow \{s \in \overline{P} \cup P'', \nexists s' \in \overline{P} \cup P'', s' \succ s\}$
$P \leftarrow ReducePop(P'' \setminus \overline{P}, PopulationSize)$
$P' \leftarrow Popselection(PopulationSize, P, \overline{P})$
$P'' \leftarrow \emptyset$

Main loop

while *StoppingCondition* not met **do**
Crossover
while $|P'| > 1$ **do**
$(s_1, s_2) \leftarrow RandomSelect(s \in P'), s_1 \neq s_2$
$P' \leftarrow P' \setminus \{s_1, s_2\}$
if $CrossoverProb$ **then**
$(s_3, s_4) \leftarrow Crossover(s_1, s_2)$
$(sc_1, sc_2, sc_3, sc_4) \leftarrow (RepairCost(s_3), RepairCost(s_4), RepairRate(s_3), RepairRate(s_4))$
$P'' \leftarrow P'' \cup feasible(\{sc_1, sc_2, sc_3, sc_4\})$
end if
else
$P'' \leftarrow P'' \cup \{s_1, s_2\}$
end else
end while
$\overline{P'} \leftarrow \{s \in \overline{P} \cup P, \nexists s' \in \overline{P} \cup P, s' \succ s\}$
$P'' \leftarrow \{P'' \cup P'\}$
$P \leftarrow \emptyset$

Mutation
$T \leftarrow P''$
while $|T| \neq \emptyset$ **do**
$s_1 \leftarrow RandomSelect(s \in P'')$
$T \leftarrow \{T \setminus s_1\}$
$(s_1, x_{i_1 j_1}, x_{i_2 j_2}) \leftarrow Mutation(s_1, MutationProb)$
$(sm_1, sm_2) \leftarrow feasible\{RepairCostMut(s_1, x_{i_1 j_1}, x_{i_2 j_2}), RepairRateMut(s_1, x_{i_1 j_1}, x_{i_2 j_2})\}$
$P'' \leftarrow P'' \cup feasible(\{sm_1, sm_2\})$
end while

Path-relinking
$\overline{P} \leftarrow \{s \in \overline{P} \cup P'', \nexists s' \in \overline{P} \cup P'', s' \succ s\}$
$PR() \leftarrow \emptyset$
while $|PR()| < |\overline{P}| - 1$ **do**
$(s_1, s_2) \leftarrow RandomSelect(s \in \overline{P}, s \notin PR()), s_1 \neq s_2$
$PR() \leftarrow PR() \cup (s_1, s_2)$
$P'' \leftarrow P'' \cup \{PathRelinking(s_1, s_2)\}$
end while

Update the archive and the population
$\overline{P} \leftarrow \{s \in \overline{P} \cup P'', \nexists s' \in \overline{P} \cup P'', s' \succ s\}$
$P \leftarrow ReducePop(P'' \setminus \overline{P}, PopulationSize)$
$P' \leftarrow Popselection(PopulationSize, P, \overline{P})$
$P'' \leftarrow \emptyset$
end while

return $\overline{P}$

8.3.3 Crossover

A crossover operator consists of splitting a pair of solutions, called parents and associating this partial solutions to produce further solutions, called offspring.
A one-point crossover is selected for A-SPEA2. Each solution is splitted into two parts. Then two new offspring are built by assembling two complementary parts, each of them comes from each parent.
As no crossover operator ensures to produce feasible offspring, the choice of a crossover should consider supplementary efforts to repair them. For this purpose, two main approaches can be considered:

- a crossover that handles sets of areas *[cross1]*
  In this approach, two complementary set of areas are used to form a new offspring. Each set consists of an area and the assignment variables related to this area.

- a crossover that handles sets of sites *[cross2]*
  In this approach, two complementary set of sites are used to form a new offspring. Each set consists of a site and the assignment variables related to this site.

The first alternative *cross1* can generate solutions whose capacity constraint violations are numerous, because of the different origins of the area assignment. In the other approach *cross2*, the offspring violate no capacity constraint. For this reason, the selected crossover is based on the first approach *cross2*.
The crossover is illustrated by the figures 8.1 and 8.2.

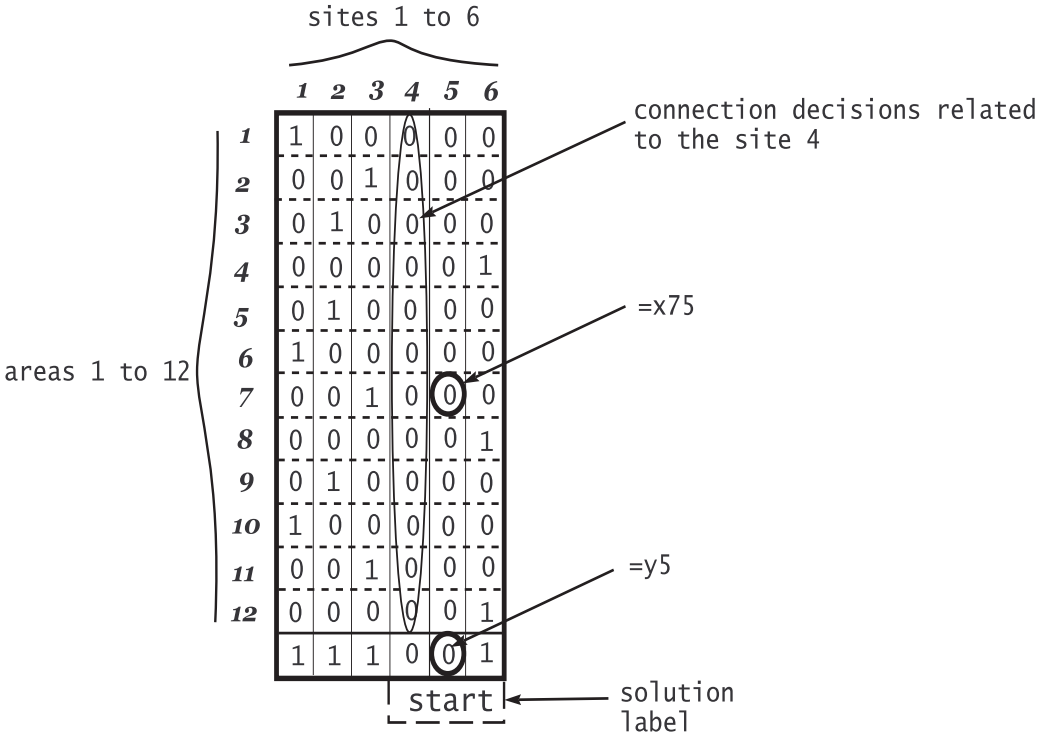


Figure 8.1: Details about the representation of a solution.

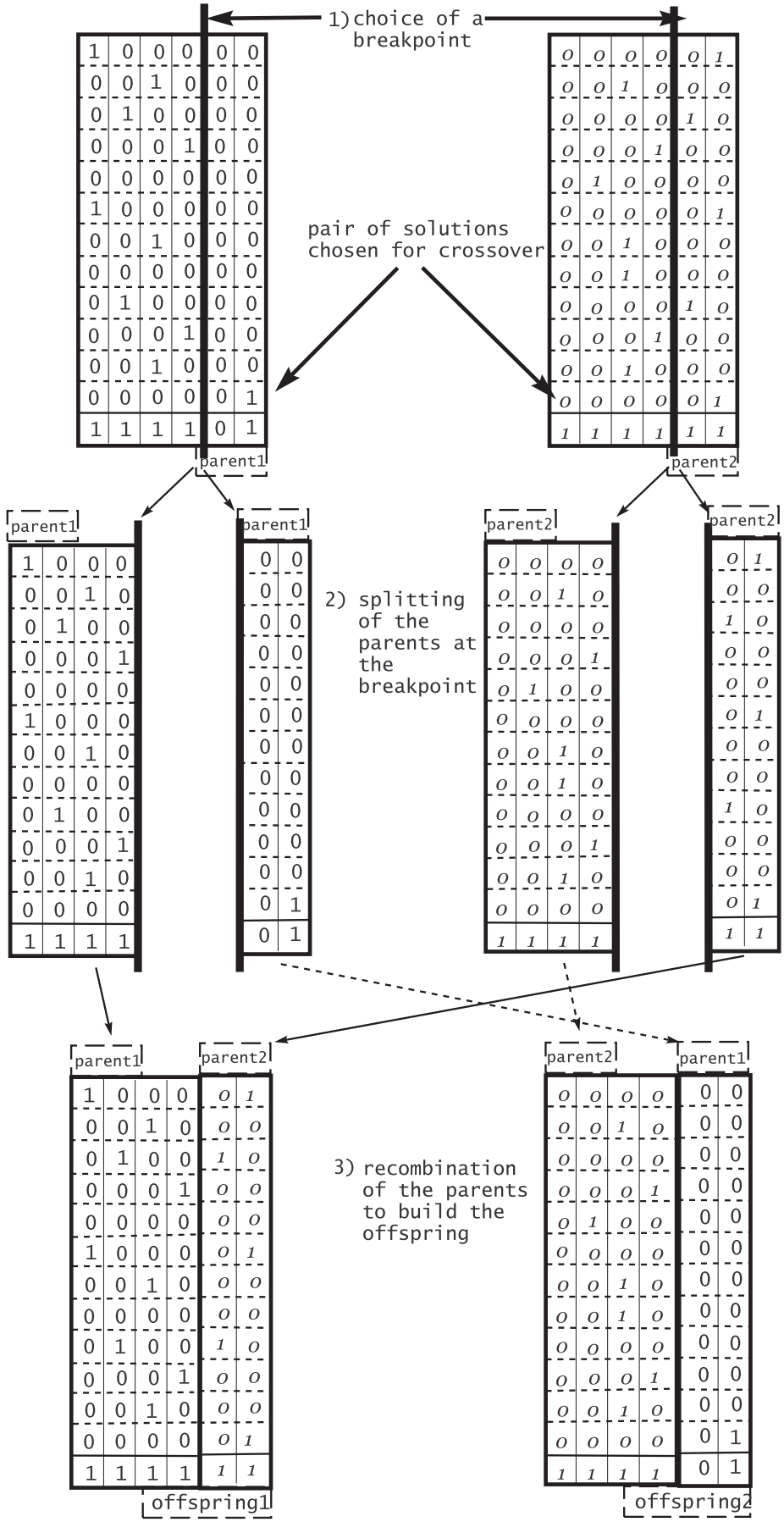


Figure 8.2: Illustration of the crossover.

The **algorithm 8.3.3** of the crossover procedure is described below.

Algorithm 8.3.3 *function Crossover(s1, s2, CrossoverProb)*

Crossover($s1$, $s2$)
Choose randomly one breakpoint B that lies between 1 and $J-1$.
- A first offspring off_1 is generated from the union of the B first sites of the solution $s1$ and the $J-B$ sites of the solution $s2$.
- A second offspring off_2 is generated from the union of the $J-B$ first sites of the solution $s1$ and the B last sites of the solution $s2$.
return (off_1, off_2)

A procedure to repair each offspring produced by an evolutionary operator is necessary to prevent from rejecting promising unfeasible offspring. This procedure is divided in two main steps:

- deleting and adding connections in order to satisfy the connectivity constraints. The constraint of capacity limit is not considered through this step. The best alternative [1] among the candidate connections is selected.
- substituting connections by other ones such that the capacity limit can be satisfied: among the connections that can be used for the substitution, the best alternative[1] is selected.

Two repairing algorithms are developed to lead the repairing efforts to each direction, the **algorithm 8.3.4** focuses on the cost, and the **algorithm 8.3.5** focuses on the rate ρ.

Algorithm 8.3.4 *function RepairCost(s)*

for each demand area i **do**
If the number of connections to each area s is greater than N:
assess the cost given by the deletion of each of the connections connected to this area i and select the best deletion alternative.
end for

for each demand area i **do**
If the number of connections to each area i is lower than N:
assess the cost given by the addition of each of the connections not connected to this area i and select the best additionn alternative.
end for

for each site j **do**
if the capacity of the site j is exceeded **then**
for each area i **do**
if the area i is connected to the site j **then**
If possible, connect the area i to another site $j' \neq j$ such that:
- the site j' can satisfy the additional demand $\frac{d_i}{N}$
- if several sites j' satisfies the first condition, select the site that favors the cost
end if
end for
end if
end for

return s

[1] The best alternative is deduced straighforwardly: with regard to an objective function (cost or rate ρ), the corresponding objective function value of each candidate is calculated while considering the impact of this candidate on the solution. The best alternative is given by the lowest calculated objective function value. If several candidates gives this value, the first candidate is chosen.

Algorithm 8.3.5 *function RepairRate(s)*

for each demand area i **do**
If the number of connections to each area s is greater than N:
assess the rate given by the deletion of each of the connections connected to this area i and select the best deletion alternative.
end for

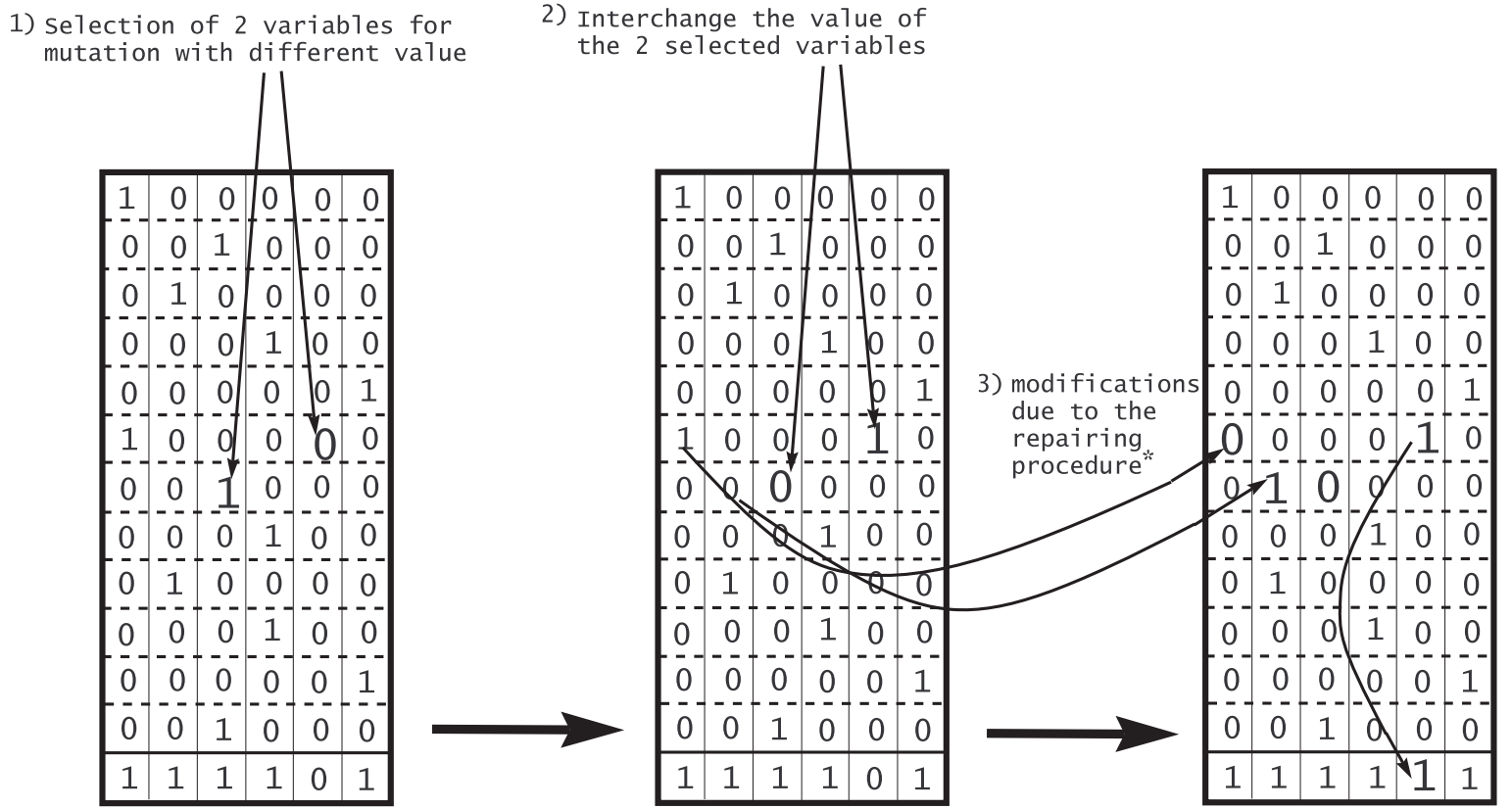
for each demand area **do**
If the number of connections to each area i is lower than N:
assess the rate given by the addition of each of the connections not connected to this area i and select the best additionn alternative.
end for

for each site j **do**
if the capacity of the site j is exceeded **then**
for each area i **do**
if the area i is connected to the site j **then**
If possible, connect the area i to another site $j' \neq j$ such that:
- the site j' can satisfy the additional demand $\frac{d_i}{N}$
- if several sites j' satisfies the first condition, select the site that favors the rate ρ
end if
end for
end if
end for

return s

8.3.4 Mutation

The mutation operator operates a random modification of a solution that is applied with a probability $MutationProb$ to the current population. This operator generates one additional solution by switching randomly two binary variables.
In our case, only the assignment variables are subject to a mutation as closing or adding a site would require a much greater effort to make the offspring feasible. Usually one binary decision variable is changed through a mutation. As opening a connection can not be done without closing another one and conversely, the mutation operator built from this problem changes simultaneously two decisions variables : one equal to 1 is set to 0 and another one equal to 0 is set to 1.
Afterwards, a repairing algorithm is developed to prevent from rejecting a promising unfeasible solution. The figure 8.3 illustrates how the mutation works.



*x_{61} is set to 0 and x_{72} to 1 to respect the constraint $\sum_{j=1,\ldots,J} x_{ij} = N$
y_5 is set to 1 to respect the constraint $x_{ij} \leq y_j$

Figure 8.3: Illustration of the mutation.

The complete mutation algorithm is detailed by the **algorithm 8.3.6**.

Algorithm 8.3.6 *function Mutation(s, MutationProb)*

Choose randomly 2 assignment variables x_{ij} and x_{kl} of the solution s' ($(i,k) \in 1,,...,,I$ and $(j,l) \in 1,,...,,J$).

- x_{ij} is substituted by the closest following assignment variable equal to 1, $x_{i'j'}$.
- x_{kl} is substituted by the closest following assignment variable equal to 0, $x_{k'l'}$.

$x_{i'j'}$ is set to 0 and $x_{k'l'}$ is set to 1.

Mark the variables $x_{i'j'}$ and $x_{k'l'}$ as forbidden in modification from now on.

return $s, x_{i'j'}, x_{k'l'}$.

After choosing two variables that are used to process the mutation, a repairing method is used to make the offspring feasible and different from its parents as the repairing method may result in the original solution. The repairing procedure is similar to that developed in the previous subsection.
There is only one modification that is an additional constraint: any change in the assignment variables is allowed as long as the value of the variables $x_{i'j'}$ and $x_{k'l'}$ is not changed. This constraint is added at the beginning of these repairing **algorithms 8.3.7 and 8.3.8**.

Algorithm 8.3.7 *function RepairCostMut(s, x_{ij}, x_{kl})*

Mark the variables x_{ij} and x_{kl} as forbidden in modification.

for each demand area i **do**

If the number of connections to each area s is greater than N:

assess the cost given by the deletion of each of the connections connected to this area i and select the best deletion alternative.

end for

for each demand area **do**

If the number of connections to each area i is lower than N:

assess the cost given by the addition of each of the connections not connected to this area i and select the best additionn alternative.

end for

for each site j **do**

if the capacity of the site j is exceeded **then**

for each area i **do**

if the area i is connected to the site j **then**

If possible, connect the area i to another site $j' \neq j$ such that:

- the site j' can satisfy the additional demand $\frac{d_i}{N}$
- if several sites j' satisfies the first condition, select the site that favors the cost

end if

end for

end if

end for

return s

Algorithm 8.3.8 *function RepairRateMut(s, x_{ij}, x_{kl})*

Mark the variables x_{ij} and x_{kl} as forbidden in modification.

for each demand area i **do**
If the number of connections to each area s is greater than N:
assess the rate given by the deletion of each of the connections connected to this area i and select the best deletion alternative.
end for

for each demand area **do**
If the number of connections to each area i is lower than N:
assess the rate given by the addition of each of the connections not connected to this area i and select the best additionn alternative.
end for

for each site j **do**
if the capacity of the site j is exceeded **then**
for each area i **do**
if the area i is connected to the site j **then**
If possible, connect the area i to another site $j' \neq j$ such that:
- the site j' can satisfy the additional demand $\frac{d_i}{N}$
- if several sites j' satisfies the first condition, select the site that favors the rate ρ
end if
end for
end if
end for

return s

8.3.5 Path-relinking

Fundamental principle

Path-relinking (PR) is used as an intensification mechanism. It is a way to explore trajectories between elite solutions obtained by the MOEA process. Starting with a pair of elite solutions, PR builds a path in the search space that joins the pair of elite solutions. The purpose of this method is to explore agressively the neighboorhood of the set of non-dominated solutions to complete it. The figures 8.4 to 8.6 illustrates this principle.

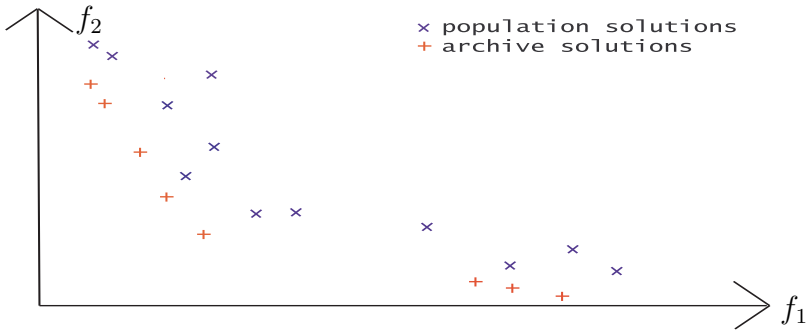


Figure 8.4: Illustration before using PR

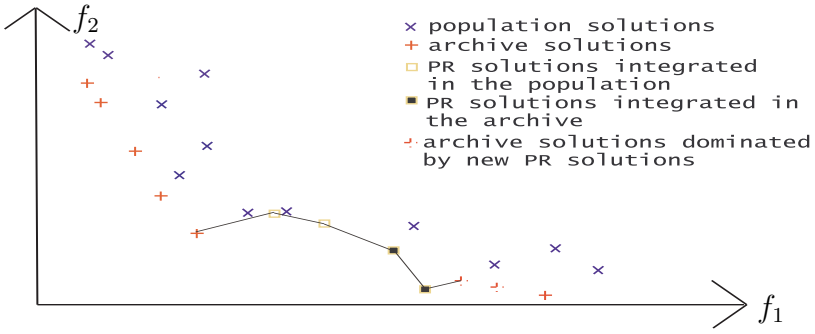


Figure 8.5: Path-relinking process

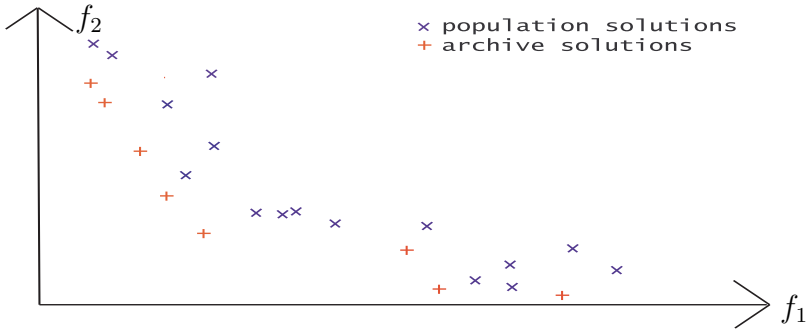


Figure 8.6: Illustration after using PR

To generate paths, moves are selected to introduce attributes in the current solution that are present in the elite guiding solution. The generated path is unknown before the PR process. For that reason, no new elite solutions may be produced but any new feasible solution produced by the PR process and dominated is used to complete the current population. If necessary the solutions produced by the PR process are repaired by using both repairing functions $RepairCost(s)$ and $RepairRate(s)$. Consequently the PR process leads to two paths betweens each pair of selected solutions, one that favors the cost and one that favors the rate ρ.

Algorithm

Basically the PR process consists of interchanging decision variables of the solution *start* to make progressively this solution equal to the solution *end*.

Considering a pair of non-dominated solutions, the solution whose number of open sites is the lower is denoted *start* and the other solution is denoted *end*. The PR-relinking process developed for our purpose links two non-dominated solutions by going from the solution *start* to the other solution *end*. Each time a site j of the solution *start* is open, this solution *start* heritates the connection decisions related to this site j.
Three basic movements -***mvt1***, ***mvt2*** and ***mvt3***- are selected for the PR process:

> ***mvt1(j,start,end)*** Each open site j of the solution *start* and the connections related to this site are closed if the same site j of the solution *end* is closed. This site j is substituted by the smallest site:
>
> $$\begin{cases} \text{whose capacity is greater than the site just closed} \\ \text{that is closed in the solution } start \end{cases}$$
>
> If no site verifies the two conditions above, the closed site j is substituted by the open site in the solution *end*:
>
> $$\begin{cases} \text{whose capacity is the greatest} \\ \text{that is closed in the solution } start \end{cases}$$
>
> ***mvt2(j,start,end)*** Each closed site j of the solution *start* are opened if the same site j of the solution *end* is open.
>
> ***mvt3(j,start,end)*** Each open site j of the solution *start* that is also open in the solution *end* is kept open and only the connections of this site j are changed to be the same as that of the site j in the solution *end*.

The four figures 8.7 to 8.10 illustrate the three basic movements used in the PR process.

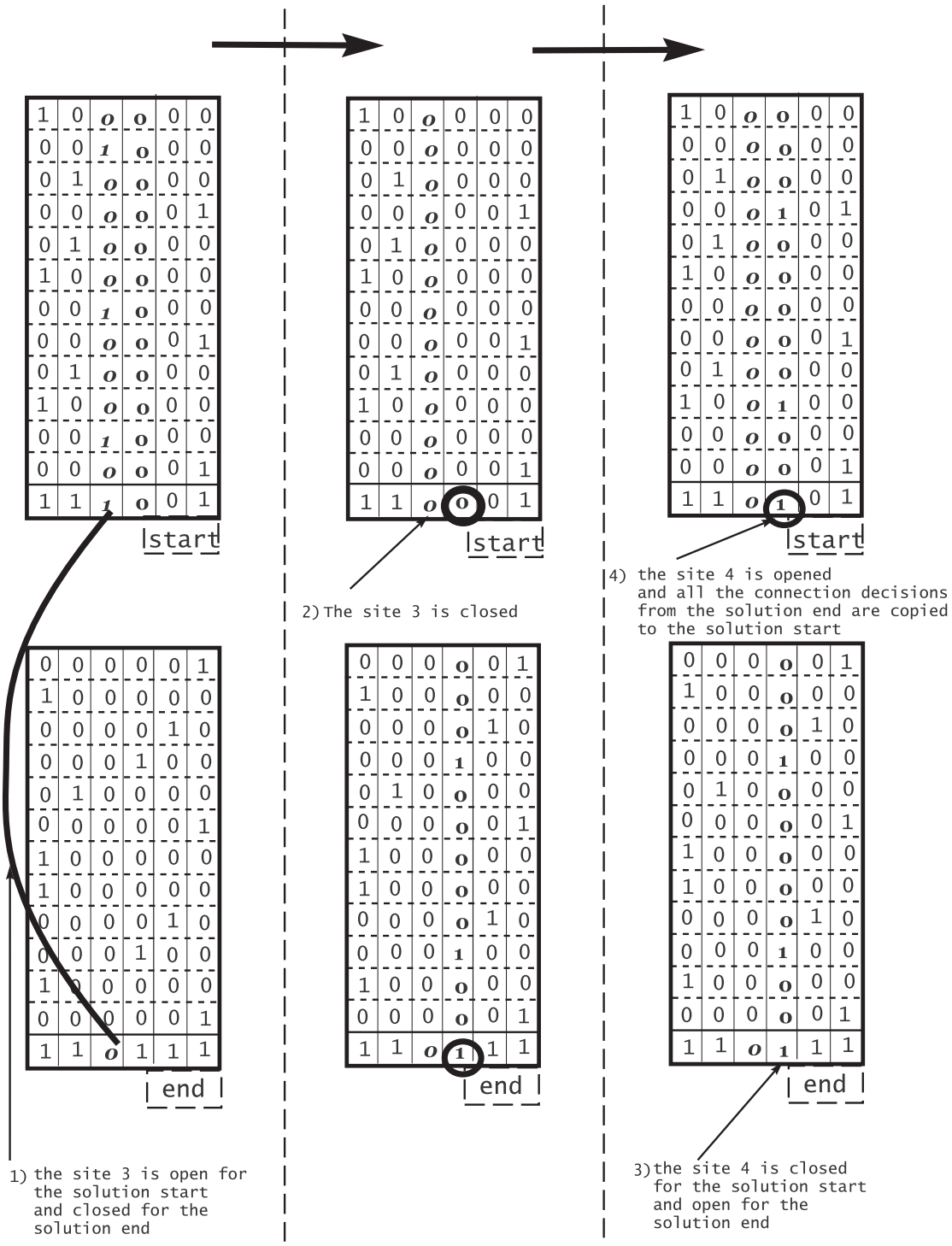


Figure 8.7: Illustration of the movement ***mvt1***.

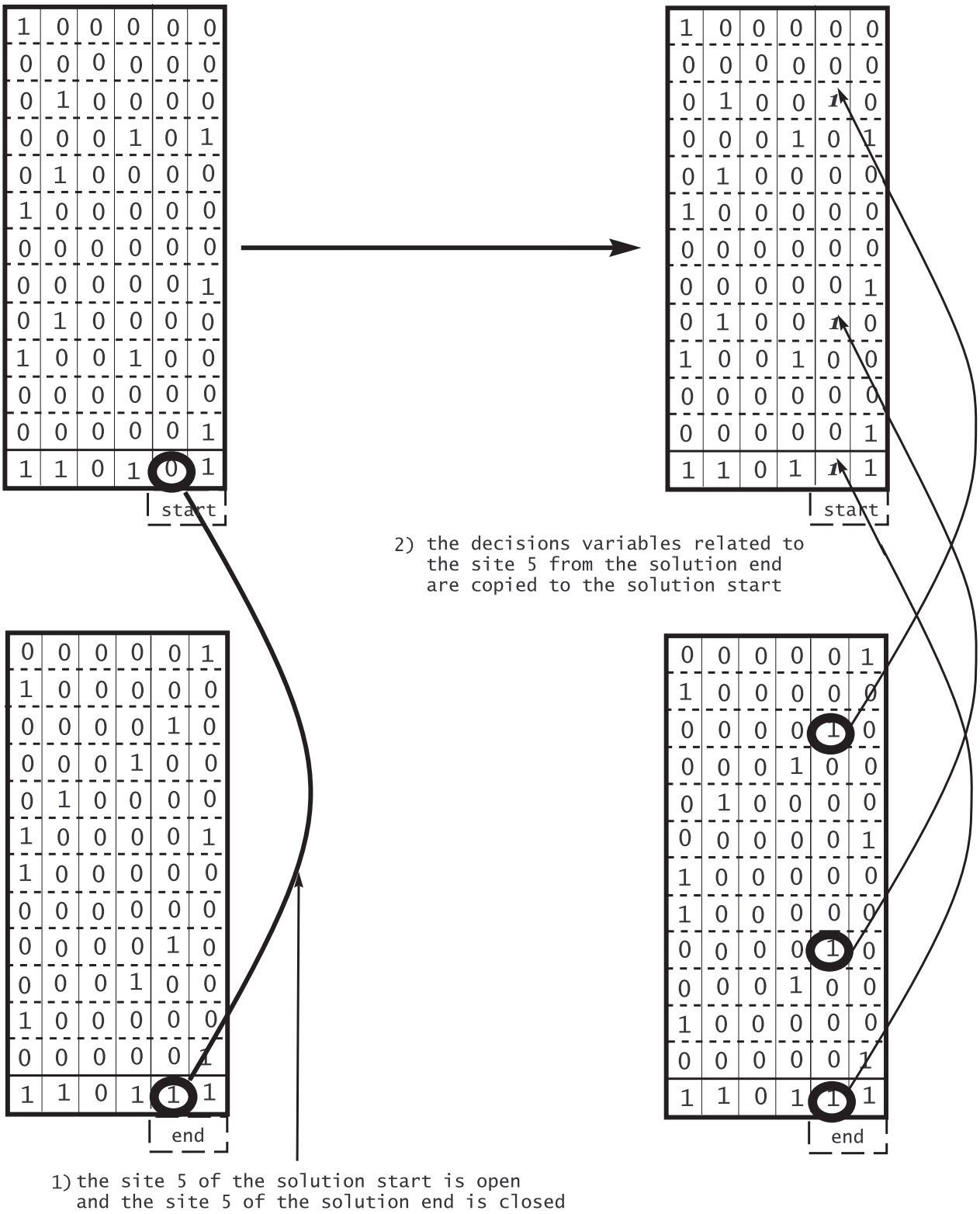


Figure 8.8: Illustration of the movement ***mvt2***.

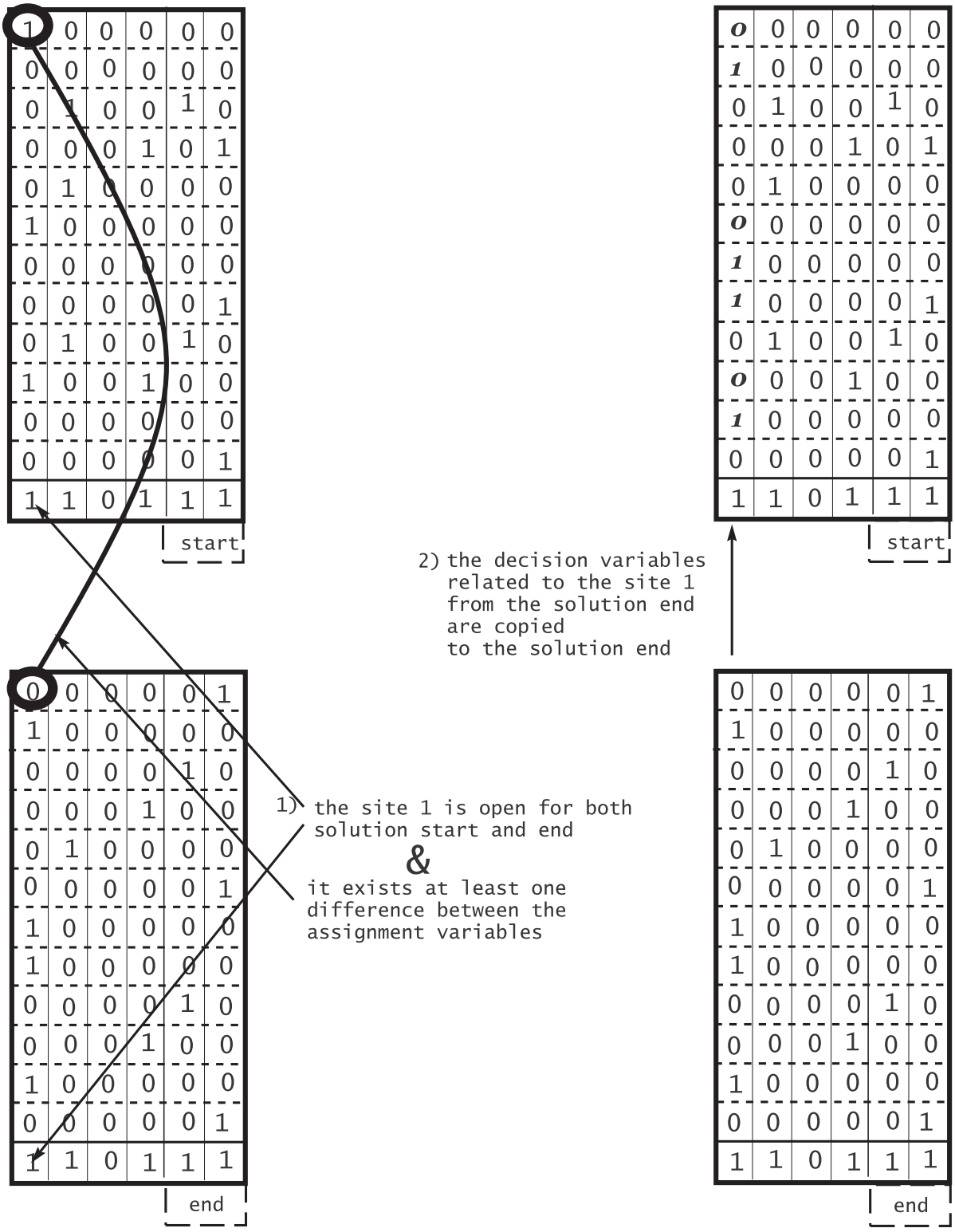


Figure 8.9: Illustration of the movement **mvt3**.

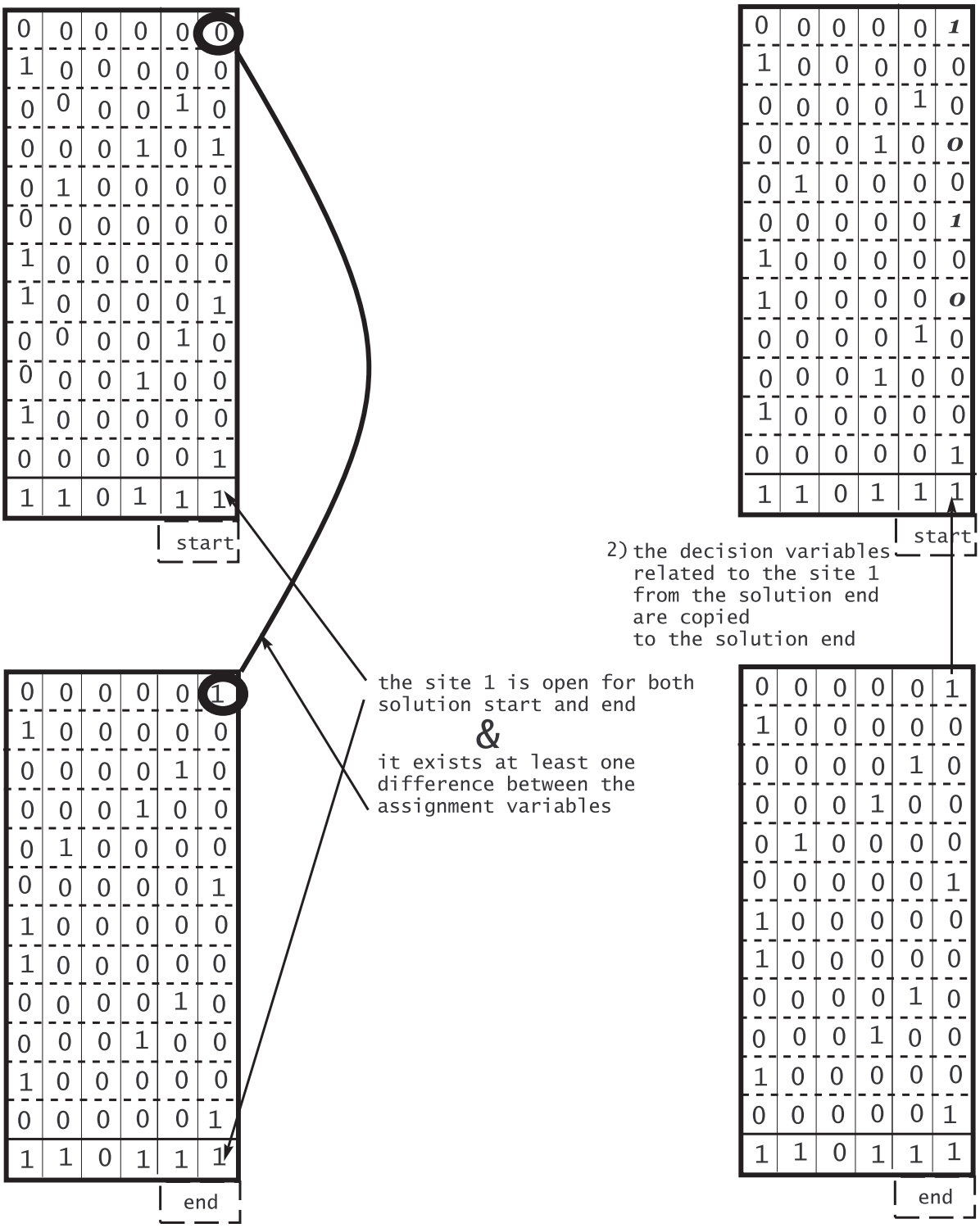


Figure 8.10: Illustration of the movement **mvt3** (continued).

Chapter 9

Computational results of the A-SPEA2

9.1 Experimentation protocol

The given limit time to run the MOEA was set to 3 hours. With respect to this time constraint, a previous period of experimental test has allowed to validate the most suitable parameters for the MOEA.
Concerning the initial population, dividing the search space in $N = 30$ intervals give the best results. It could have been possible to give the same amount of time $TOTALTIME$ whatever the size of the instance to solve. However it was preferred to give an amount of time that increases with the size of the instance as follows:

- $TOTALTIME(I_1 = 100, 100) = 2000$ for $I = 100$ and $J = 100$
- $TOTALTIME(I_2 = 75, 75) = 1500$ for $I = 75$ and $J = 75$
- $TOTALTIME(I_3 = 50, 75) = 1000$ for $I = 50$ and $J = 50$
- $TOTALTIME(I_4 = 25, 10) = 500$ for $I = 25$ and $J = 10$

The parameter $TOTALTIME$ for the other instances is deduced as follows ($i = 1, ..., 3, J < I_i$):

$$TOTALTIME(I_i, J) = TOTALTIME(I_{i+1}, I_{i+1}) + (TOTALTIME(I_i, I_i) - TOTALTIME(I_{i+1}, I_{i+1})) * \frac{J}{I_i}$$

Our experimental tests have shown that the early integration of the greedy solutions leads to an archive whose solutions with a relatively low cost are dominated by the solutions of the archive obtained from the same initial population without the greedy solutions. This is due to the relatively high cost of the greedy solutions. This behavior incited us to delay the integration of the greedy solutions. Our experimental test have shown that the best results are obtained by integrating them at the middle of the evolutionary process.
During the experimental period, the greedy population and the MMXCplex population given by the function *Build(TOTALTIME,P)* were confronted through the SPEA engine and the obtained archive were compared. As expected the greedy population results in a archive that is dominated by the other archive. The next figure 9.1 illustrates this set of observations.

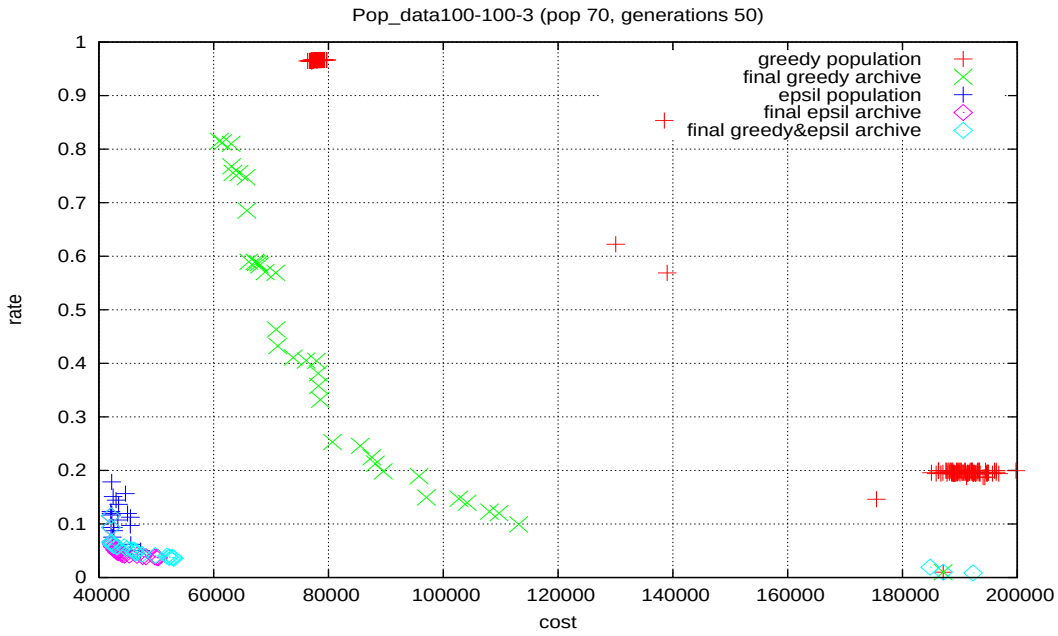


Figure 9.1: Comparison of archives obtained from different intial populations for the problem `BiFLP100-100-3.VC.1`

Several crossover probabilities were tested and compared by the archives. The best level was 60% whatever the size of the problem. Above 60%, the given archive was very similar and a higher crossover probability leads to an increase of computational time. For these two reasons, the value 60% is considered as the best one.
The mutation operator does not give better results whatever the probability of mutation and the size of problem. This operator allows to improve the population by improving its diversity. However these additional members do not affect the archive. For this reason, the mutation probability is set to 0%.
The number of generations is fixed to 100 as no real improvement is measured above this limit.
The PR process is aimed to complete a mature population. For this reason it is activated from the beginning of the evolutionary process. Experimental tests have proved that delaying the activation of the PR process is not relevant.

9.2 Results

The MOEA was applied on all the instances with a number of sites that lies between 25 and 100. The computational results are available in the table 9.1 for the problem with 25 sites, the table 9.2 for the instances with 50 sites, the tables 9.3 and 9.4 for the instances with 75 sites and the tables 9.5 and 9.6 for the instances with 100 sites.
To illustrate the evolution of the initial population to the final archive, a restrited set of results were plotted for the following problems: `BiFLP25-10-N.VC.1`, `BiFLP50-50-N.VC.1`, `BiFLP75-75-N.VC.1` and `BiFLP100-100-N.VC.1` with $N \in \{1, 2, 3\}$. Refer to the figures 9.2 to 9.5.
Concerning the problems `BiFLP25-10-N.VC.1`, the extreme points were evaluated within a limited amount of time. The solutions that give the best found rate ρ and the best found cost within 45 hours have been also added to the figure 9.2.

	Initial population							path-relinking activated		
instance	building time limit (sec)	CPU_t (sec)	# exact solutions	initial population size	inserted greedy	CPU_t (sec) for MOEA	final archive size	# paths produced by PR	CPU_t (sec) for MOEA	final archive size
BiFLP25-10.1.VC.1	500	437	1	31 (70)	0	6	8	327	6	17
BiFLP25-10.1.VC.2	500	22	29	34 (70)	0	4	12	150	4	12
BiFLP25-10.1.VC.3	500	287	17	29 (70)	0	5	12	433	6	14
BiFLP25-10.1.VC.4	500	301	17	29 (70)	0	5	14	239	6	10
BiFLP25-10.2.VC.1	500	115	26	34 (70)	0	6	15	207	6	14
BiFLP25-10.2.VC.2	500	6	29	34 (70)	1	4	17	343	4	19
BiFLP25-10.2.VC.3	500	55	28	34 (70)	0	5	12	173	5	12
BiFLP25-10.2.VC.4	500	55	28	34 (70)	0	5	12	173	5	12
BiFLP25-10.3.VC.1	500	60	28	34 (70)	0	5	30	612	5	28
BiFLP25-10.3.VC.2	500	8	29	34 (70)	1	4	17	253	4	16
BiFLP25-10.3.VC.3	500	110	27	34 (70)	0	6	25	540	6	25
BiFLP25-10.3.VC.4	500	110	27	34 (70)	0	6	25	540	7	25

Table 9.1: Computational results for the MOEA for the instances `BiFLP25-10-N.VC.x`.

	Initial population							path-relinking activated		
instance	building time limit (sec)	CPU_t (sec)	# exact solutions	initial population size	inserted greedy	CPU_t (sec) for MOEA	final archive size	# paths produced by PR	CPU_t (sec) for MOEA	final archive size
BiFLP50-25.1.VC.1	750	640	6	24 (70)	0	37	6	48	39	7
BiFLP50-25.1.VC.2	750	570	10	25 (70)	0	38	5	33	39	7
BiFLP50-25.1.VC.3	750	543	14	33 (70)	0	36	18	417	42	19
BiFLP50-25.1.VC.4	750	696	1	31 (70)	0	33	11	152	36	8
BiFLP50-25.2.VC.1	750	717	9	33 (70)	0	61	13	116	61	11
BiFLP50-25.2.VC.2	750	402	20	31 (70)	0	42	13	185	45	14
BiFLP50-25.2.VC.3	750	368	24	34 (70)	0	31	14	315	39	18
BiFLP50-25.2.VC.4	750	471	16	34 (70)	0	36	17	367	42	17
BiFLP50-25.3.VC.1	750	222	28	34 (70)	0	39	17	382	47	19
BiFLP50-25.3.VC.2	750	163	28	34 (70)	1	41	50	1106	64	48
BiFLP50-25.3.VC.3	750	198	26	34 (70)	0	32	28	663	43	29
BiFLP50-25.3.VC.4	750	199	26	34 (70)	0	33	18	390	41	20
BiFLP50-50.1.VC.1	1000	732	3	19 (70)	0	141	15	447	195	14
BiFLP50-50.1.VC.2	1000	663	3	11 (70)	0	110	8	216	134	8
BiFLP50-50.1.VC.3	1000	597	3	15 (70)	0	148	11	428	158	14
BiFLP50-50.1.VC.4	1000	670	3	14 (70)	0	116	10	547	187	17
BiFLP50-50.2.VC.1	1000	932	1	31 (70)	1	216	18	643	243	28
BiFLP50-50.2.VC.2	1000	770	3	19 (70)	0	200	15	218	215	11
BiFLP50-50.2.VC.3	1000	936	2	29 (70)	0	142	13	268	170	12
BiFLP50-50.2.VC.4	1000	869	2	27 (70)	0	209	10	214	210	13
BiFLP50-50.3.VC.1	1000	1004	1	33 (70)	1	223	28	845	321	35
BiFLP50-50.3.VC.2	1000	703	3	13 (70)	0	193	3	282	216	10
BiFLP50-50.3.VC.3	1000	937	1	31 (70)	0	192	19	381	232	17
BiFLP50-50.3.VC.4	1000	934	2	29 (70)	0	220	13	480	213	17

Table 9.2: Computational results for the MOEA for the instances `BiFLP50-25-N.VC.x` and `BiFLP50-50-N.VC.x`.

	Initial population							path-relinking activated		
instance	building time limit (sec)	CPU_t (sec)	# exact solutions	initial population size	inserted greedy	CPU_t (sec) for MOEA	final archive size	# paths produced by PR	CPU_t (sec) for MOEA	final archive size
BiFLP75-25.1.VC.1	1166	1070	2	29 (70)	0	71	15	444	77	16
BiFLP75-25.1.VC.2	1166	953	2	21 (70)	0	63	8	229	77	11
BiFLP75-25.1.VC.3	1166	1108	1	32 (70)	0	91	10	197	98	14
BiFLP75-25.1.VC.4	1166	1106	2	29 (70)	0	91	11	150	103	11
BiFLP75-25.2.VC.1	1166	1107	2	31 (70)	0	98	9	104	115	8
BiFLP75-25.2.VC.2	1166	1108	1	32 (70)	0	57	21	690	94	27
BiFLP75-25.2.VC.3	1166	1026	12	34 (70)	0	72	17	332	88	15
BiFLP75-25.2.VC.4	1166	621	27	34 (70)	1	64	17	516	80	25
BiFLP75-25.3.VC.1	1166	781	16	34 (70)	1	113	19	371	141	21
BiFLP75-25.3.VC.2	1166	831	20	34 (70)	0	50	22	335	64	18
BiFLP75-25.3.VC.3	1166	301	26	34 (70)	1	71	31	654	95	31
BiFLP75-25.3.VC.4	1166	1144	1	34 (70)	0	60	18	429	80	22
BiFLP75-50.1.VC.1	1333	128	20	21 (70)	1	332	1	35	410	2
BiFLP75-50.1.VC.2	1333	1198	3	14 (70)	0	274	11	543	367	15
BiFLP75-50.1.VC.3	1333	707	3	9 (70)	0	221	14	494	303	10
BiFLP75-50.1.VC.4	1333	894	4	8 (70)	1	124	5	177	153	7
BiFLP75-50.2.VC.1	1333	1292	2	30 (70)	0	316	14	587	399	17
BiFLP75-50.2.VC.2	1333	1162	3	23 (70)	1	390	5	57	489	6
BiFLP75-50.2.VC.3	1333	1163	3	23 (70)	1	301	14	127	350	8
BiFLP75-50.2.VC.4	1333	1303	1	32 (70)	0	155	13	225	192	12
BiFLP75-50.3.VC.1	1333	1133	21	32 (70)	1	348	21	606	495	20
BiFLP75-50.3.VC.2	1333	1291	3	29 (70)	0	380	10	136	380	11
BiFLP75-50.3.VC.3	1333	1338	1	33 (70)	1	278	17	621	325	19
BiFLP75-50.3.VC.4	1333	1342	1	33 (70)	1	331	20	591	430	28

Table 9.3: Computational results for the MOEA for the instances `BiFLP75-25-N.VC.x` and `BiFLP75-50-N.VC.x`.

	Initial population							path-relinking activated		
instance	building time limit (sec)	CPU_t (sec)	# exact solutions	initial population size	inserted greedy	CPU_t (sec) for MOEA	final archive size	# paths produced by PR	CPU_t (sec) for MOEA	final archive size
BiFLP75-75.1.VC.1	1500	914	4	7 (70)	1	354	14	447	580	17
BiFLP75-75.1.VC.2	1500	1015	4	7 (70)	1	279	9	303	495	8
BiFLP75-75.1.VC.3	1500	832	3	6 (70)	1	1093	17	849	2055	25
BiFLP75-75.1.VC.4	1500	832	3	8 (70)	1	444	11	322	425	13
BiFLP75-75.2.VC.1	1500	1363	3	24 (70)	0	407	7	169	427	9
BiFLP75-75.2.VC.2	1500	1065	2	11 (70)	0	416	2	3	913	3
BiFLP75-75.2.VC.3	1500	1031	3	16 (70)	1	567	30	952	1271	29
BiFLP75-75.2.VC.4	1500	1237	2	19 (70)	1	514	12	426	735	15
BiFLP75-75.3.VC.1	1500	1169	1	20 (70)	1	955	18	675	933	20
BiFLP75-75.3.VC.2	1500	1186	1	20 (70)	1	938	28	533	1367	24
BiFLP75-75.3.VC.3	1500	1392	1	21 (70)	0	815	22	741	811	24
BiFLP75-75.3.VC.4	1500	935	1	6 (70)	1	641	8	480	1203	15

Table 9.4: Computational results for the MOEA for the instances `BiFLP75-75-N.VC.x`.

	Initial population							path-relinking activated		
instance	building time limit (sec)	CPU_t (sec)	# exact solutions	initial population size	inserted greedy	CPU_t (sec) for MOEA	final archive size	# paths produced by PR	CPU_t (sec) for MOEA	final archive size
BiFLP100-30.1.VC.1	1650	880	19	31 (70)	1	153	28	730	202	31
BiFLP100-30.1.VC.2	1650	1332	2	24 (70)	0	136	10	244	191	17
BiFLP100-30.1.VC.3	1650	1379	3	17 (70)	1	86	8	190	96	11
BiFLP100-30.1.VC.4	1650	1325	3	18 (70)	1	169	15	539	229	23
BiFLP100-30.2.VC.1	1650	635	24	34 (70)	1	165	32	841	249	36
BiFLP100-30.2.VC.2	1650	1662	1	34 (70)	1	176	10	147	208	10
BiFLP100-30.2.VC.3	1650	1599	1	32 (70)	1	237	12	188	286	13
BiFLP100-30.2.VC.4	1650	1543	1	31 (70)	0	194	18	414	252	22
BiFLP100-30.3.VC.1	1650	841	18	34 (70)	1	194	35	998	442	37
BiFLP100-30.3.VC.2	1650	783	23	34 (70)	1	181	29	551	244	28
BiFLP100-30.3.VC.3	1650	1548	2	30 (70)	1	272	10	448	321	17
BiFLP100-30.3.VC.4	1650	1654	1	34 (70)	1	255	25	706	311	26
BiFLP100-60.1.VC.1	1800	1219	3	12 (70)	1	476	24	515	436	19
BiFLP100-60.1.VC.2	1800	1340	3	7 (70)	1	396	27	569	549	16
BiFLP100-60.1.VC.3	1800	981	4	5 (70)	1	381	32	2994	3021	30
BiFLP100-60.1.VC.4	1800	1351	4	8 (70)	1	349	11	218	458	15
BiFLP100-60.2.VC.1	1800	1768	2	30 (70)	1	604	29	891	1064	35
BiFLP100-60.2.VC.2	1800	1626	3	25 (70)	1	613	26	555	900	27
BiFLP100-60.2.VC.3	1800	1591	2	27 (70)	1	505	16	546	660	20
BiFLP100-60.2.VC.4	1800	1467	2	23 (70)	1	555	21	551	925	28
BiFLP100-60.3.VC.1	1800	1544	9	33 (70)	1	691	52	1451	1553	65
BiFLP100-60.3.VC.2	1800	1359	13	32 (70)	1	622	50	1217	1324	46
BiFLP100-60.3.VC.3	1800	1524	13	33 (70)	1	617	44	1207	1086	42
BiFLP100-60.3.VC.4	1800	1567	15	33 (70)	1	690	22	549	945	24

Table 9.5: Computational results for the MOEA for the instances `BiFLP100-30-N.VC.x` and `BiFLP100-60-N.VC.x`.

	Initial population							path-relinking activated		
instance	building time limit (sec)	CPU_t (sec)	# exact solutions	initial population size	inserted greedy	CPU_t (sec) for MOEA	final archive size	# paths produced by PR	CPU_t (sec) for MOEA	final archive size
BiFLP100-80.1.VC.1	1900	1235	3	11 (70)	0	871	19	698	1439	25
BiFLP100-80.1.VC.2	1900	1091	3	11 (70)	1	685	15	456	833	17
BiFLP100-80.1.VC.3	1900	1236	4	4 (70)	1	603	21	265	589	11
BiFLP100-80.1.VC.4	1900	1236	4	4 (70)	1	603	21	265	590	11
BiFLP100-80.2.VC.1	1900	1611	2	24 (70)	1	926	18	701	1307	21
BiFLP100-80.2.VC.2	1900	1194	3	11 (70)	0	1799	11	129	2090	9
BiFLP100-80.2.VC.3	1900	1622	3	19 (70)	1	1090	31	697	1675	24
BiFLP100-80.2.VC.4	1900	1622	3	19 (70)	1	927	26	625	1464	22
BiFLP100-80.3.VC.1	1900	1786	3	17 (70)	0	2007	5	371	2785	9
BiFLP100-80.3.VC.2	1900	1812	2	29 (70)	1	2112	14	630	2068	26
BiFLP100-80.3.VC.3	1900	1748	2	26 (70)	1	1061	14	694	1789	23
BiFLP100-80.3.VC.4	1900	1748	2	26 (70)	1	1062	14	694	1789	23
BiFLP100-100.1.VC.1	2000	1644	3	12 (70)	0	997	0	2	1123	1
BiFLP100-100.1.VC.2	2000	1515	4	7 (70)	1	692	14	263	756	18
BiFLP100-100.1.VC.3	2000	581	10	12 (70)	0	1955	0	32	2308	2
BiFLP100-100.1.VC.4	2000	1240	4	6 (70)	1	770	14	310	726	18
BiFLP100-100.2.VC.1	2000	1378	3	11 (70)	1	1481	28	502	2126	18
BiFLP100-100.2.VC.2	2000	1635	2	23 (70)	0	884	10	294	1217	11
BiFLP100-100.2.VC.3	2000	1229	3	10 (70)	0	1465	20	830	2756	20
BiFLP100-100.2.VC.4	2000	1253	3	9 (70)	0	944	16	371	1099	9
BiFLP100-100.3.VC.1	2000	1728	3	15 (70)	1	2292	23	883	4382	28
BiFLP100-100.3.VC.2	2000	1715	2	24 (70)	1	2355	29	773	3901	30
BiFLP100-100.3.VC.3	2000	1455	3	14 (70)	0	1178	9	214	1800	13
BiFLP100-100.3.VC.4	2000	1782	2	23 (70)	1	2172	21	713	3726	29

Table 9.6: Computational results for the MOEA for the instances `BiFLP100-80-N.VC.x` and `BiFLP100-100-N.VC.x`.

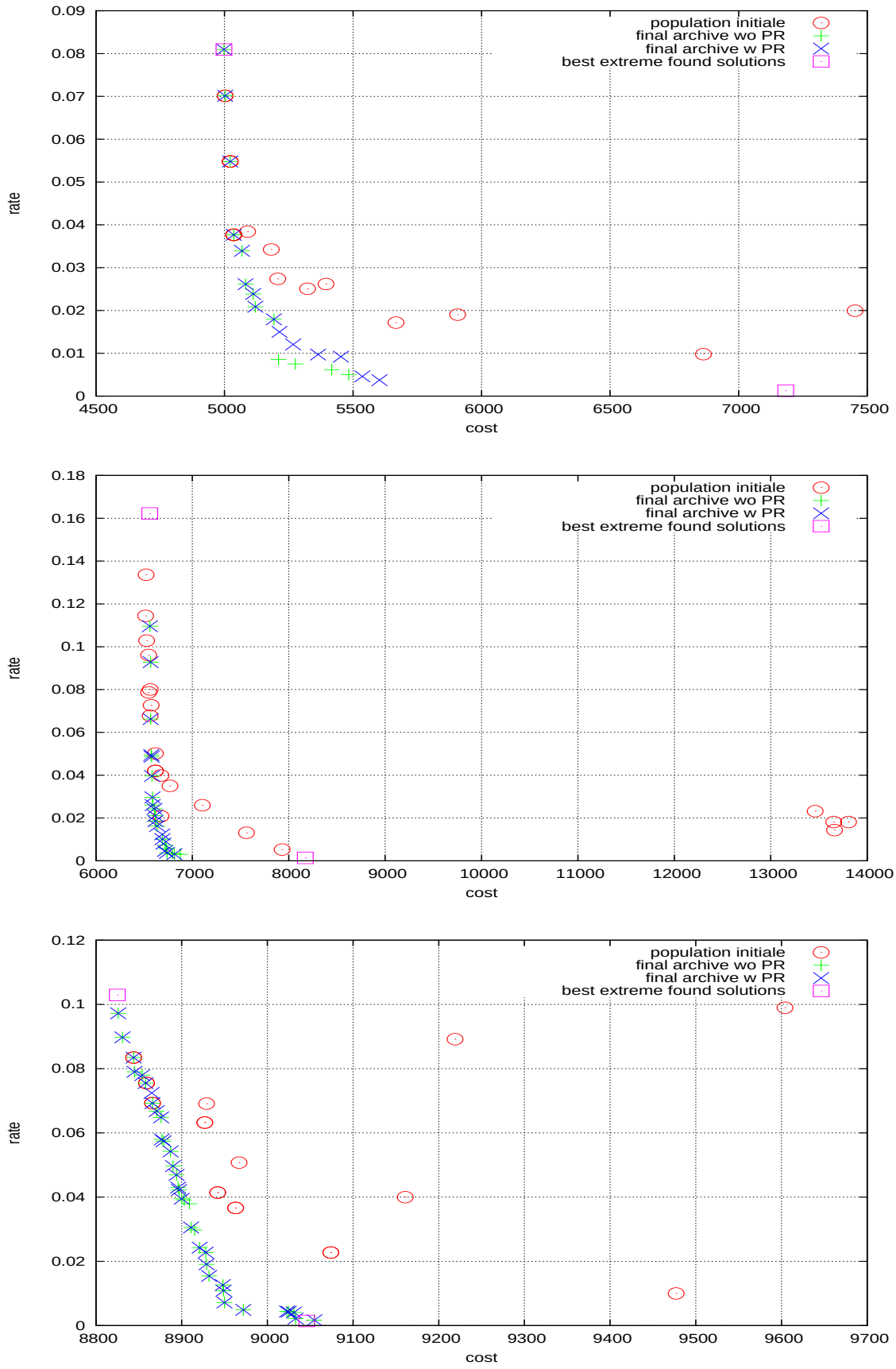
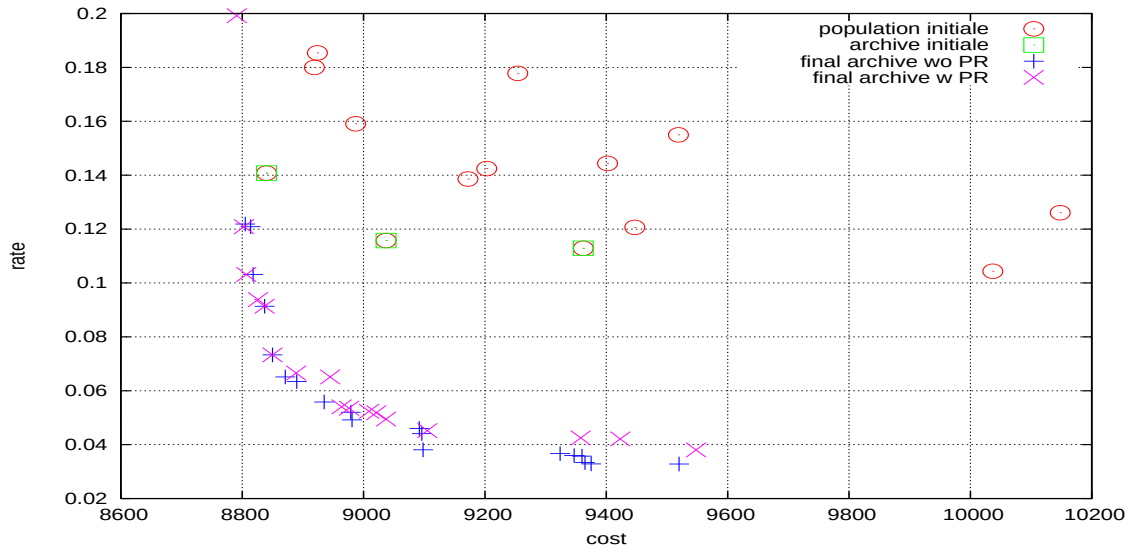
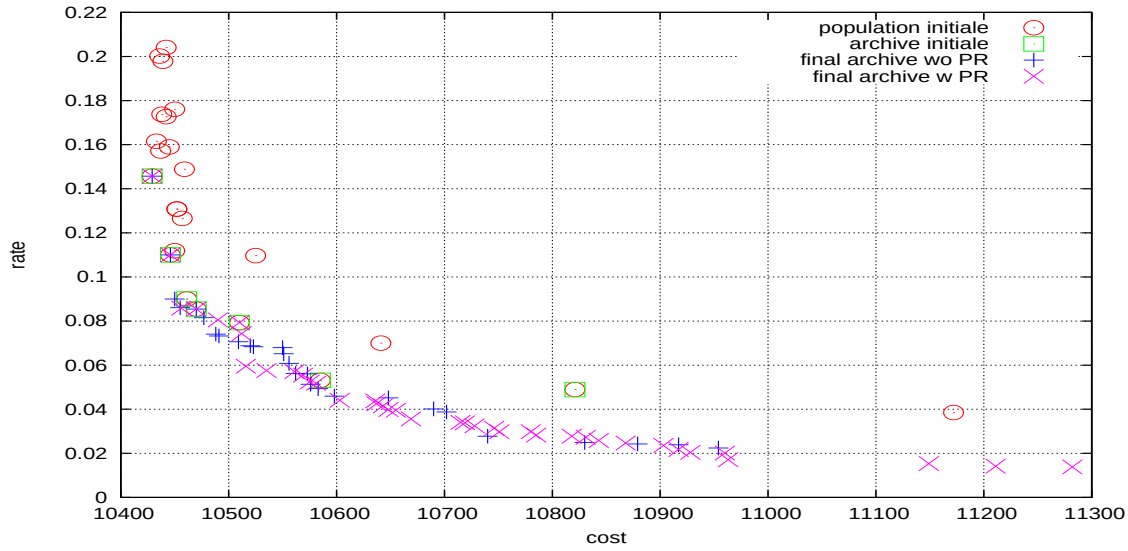


Figure 9.2: A-SPEA2 for `BiFLP25-10-N` with $N \in \{1, 2, 3\}$





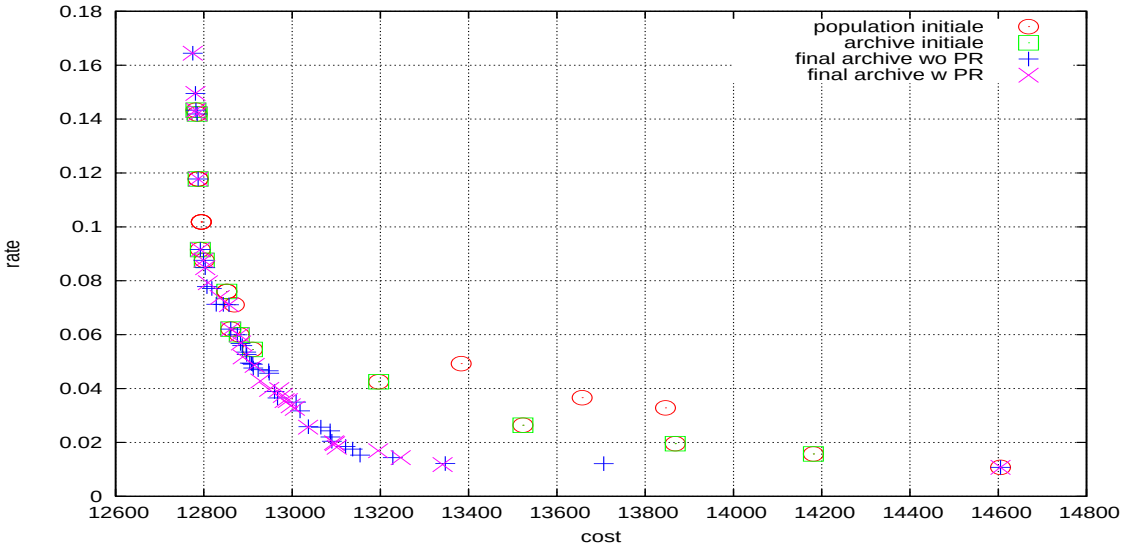


Figure 9.3: A-SPEA2 for `BiFLP50-50-N` with $N \in \{1, 2, 3\}$

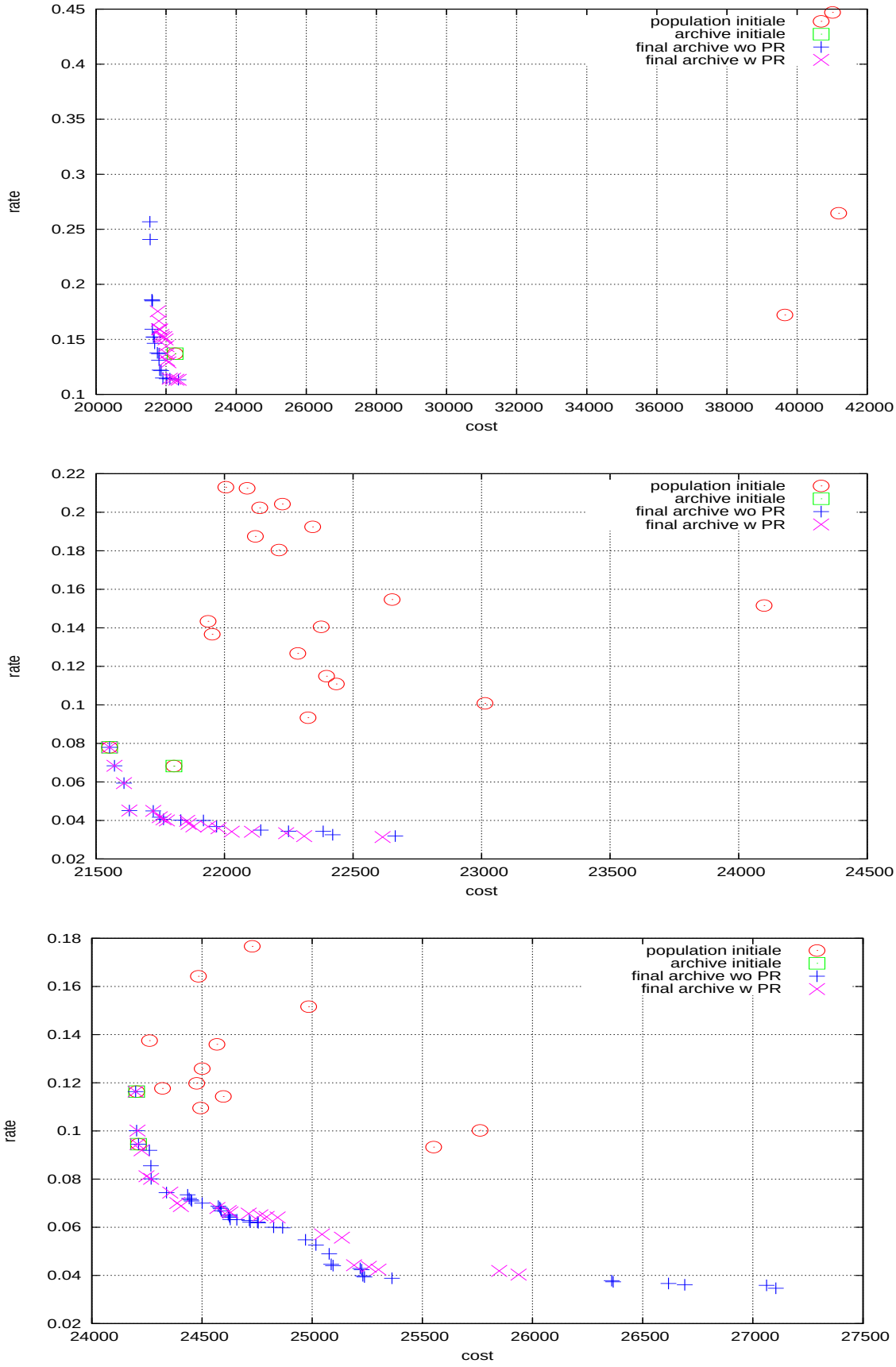


Figure 9.4: A-SPEA2 for `BiFLP75-75-N` with $N \in \{1, 2, 3\}$

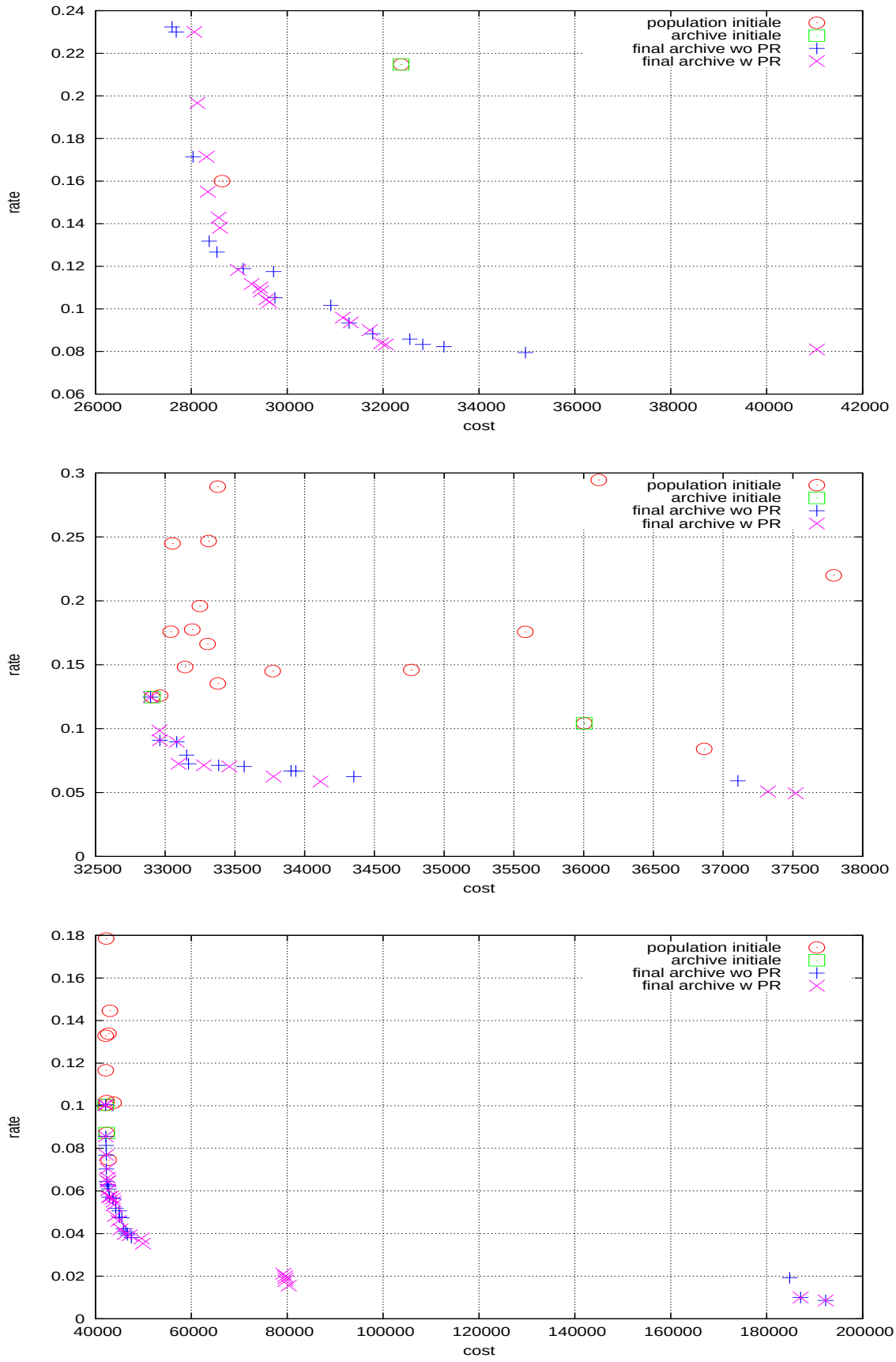


Figure 9.5: A-SPEA2 for `BiFLP100-100-N` with $N \in \{1, 2, 3\}$

9.3 Discussion

The results given by the function $Build(TOTALTIME, P)$ to generate the initial population are satisfying for all instances as the size of the initial population is higher than 12 in 117 cases out of 120. The insertion of the greedy solutions appears really significant from the instances with at least 75 sites and 50 areas. The size of the population after 2 generations between brackets appears: the crossover operator produces enough feasible offspring to complete the initial population and reach the desired size of population.
The number of exact solutions is really high, especially for the instances `biFLP25-10-N`. However this information should be considered carefully: these exact solutions are not the non-dominated points of the problem *MMX*. For instance, there is nothing that guarantees no solution with the same cost value and a lower rate exists. A comprehensive observation of the graphical results for this size of problem can confirm this statement. However a high number of exact solutions provides an initial population whose the distance along the cost axis to the set of non-dominated solutions is relatively small.
It was decided to give an increasing time limit according to the sizes of the instance. As expected, the computational time to build this population increases with the size of problem. In average, about 70% of the given time limit was used to build the initial population. However this average does not consider the instances `biFLP25-10-N` as the used amount of time lies between 1% and 80% in this case. As a consequence, the procedure to approximate this time limit gives satisfying results overall.

The MOEA process effort is given by the CPU time for MOEA. This time increases with the size of the problem from 5 seconds up to 2355 seconds. This increase of time is due to the repairing procedure incorporated to the evolutionary operators. This procedure requires more time for bigger instances due to the higher number of violated constraints and the higher number of substituting variables.

The PR process increases the MOEA process time as expected. The number of paths produced by the PR lies between 64 and 1451. There is no correlation between this number and the size of the instance. Such a correlation could be expected with the size of the final archive but this is not the case. The reason is the PR process is activated from the beginning of the MOEA process. The impact of the activation of the PR process is really limited for the instances until `biFLP75-50-N` as the CPU time of the MOEA increases by less than 35%. For the bigger instances, the computational impact becomes significant as the MOEA time can even double for the instances `biFLP100-100-3`.

The increase of the size of the final archive and the graphical results seems to show that the PR provides good results as the archive obtained with the PR operator is more complete and not dominated by the archive with the PR operator.

Chapter 10

Conclusion

The investigation of works related to our bi-objective location problem have brought out there is no work related to such a location problem regardless of the objective functions. This investigation has also displayed the existence of several interesting tracks.

In a first time, we developped an exact method for the bi-objetive location problem. The selected developed exact method for this bi-objective location problem gives right results for small-sized problems in comparison with the targeted size.

This result has proved the limit of the exact approach for the bi-objective location problem and justifies the relevance of an heuristic. Because of the multiple-objective context and the rate objective function, a metaheuristic based on an evolutionary algortithm seems appropriate as revealed by previous results given by an MOEA. Works and results presented to generate the initial population have shown there exists for this problem a wide set of approaches for this purpose. The evolutionary operators allow the evolution of the population and the increase in diversity to enrich the initial population. The Path-relinking process has led to satisfying results by increasing the number of non-dominated solutions. Overall the MOEA provides satisfying results, even for big-sized instances and without losing the control of the time limitation.

In the future, as further research we suggest :

- to improve the building method of the initial population by working on more sophisticated greedy algorithms,
- to decrease the effort to repair the solutions by introducing a way to detect the non repairable solutions,
- and to enhance the performance of the MOEA by combining another search method: the introduction of a variable neighborhood search on the non-dominated solutions could improve the final archive.

Bibliography

[ASAF99] K.S. Al-Sultan and M.A. Al-Fawzan. A tabu search approach to the uncapacitated facility location problem. *Annals of Operations Research*, pages 91–103, 1999.

[Bal66] M.L. Balinski. On finding integer solutions to linear programs. *In Proceedings IBM Scientific Computing Symposium on Combinatorial Problems*, pages 225–248, 1966.

[Bet77] J.T. Betts. An accelerated multiplier method for nonlinear programming. *Journal of Optimization Theory and Applications, Vol.21, No.2*, 1977.

[BP95] G. Bilchev and I. Parmee. Ant colony search vs. genetic algorithms. *Technical report, Plymouth Engineering Design Centre, University of Plymouth*, 1995.

[BR98] J. Brimberg and C. Revelle. A bi-objective plant location problem: cost vs. demand served. *Location Science, Vol 6*, pages 121–135, 1998.

[CFN77] G. Cornuejols, M. Fisher, and G. Nemhauser. Location of bank accounts to optimize float. *Management Science 23*, pages 789–810, 1977.

[CG02] T. Crainic and M. Gendreau. Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics, Vol 8*, pages 601–627, 2002.

[CVL02] C. Coello, D. Van Veldhuizen, and G. Lamat. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[DAPM00] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: Nsga-ii. In *PPSN*, pages 849–858, 2000.

[Dar58] C. Darwin. *On the Perpetuation of Varieties and Species by Natural Means of Selection*. 1958.

[Das95] M.S. Daskin. *Network and Discrete Location: Models, Algorithms and Applications*. John Wiley and Sons, Inc., New York, 1995.

[Deb02] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. WILEY, pages 245-253, 2002.

[DGD05] X. Delorme, X. Gandibleux, and F. Degoutin. Evolutionary, constructive and hybrid procedures for the biobjective set packing problem. *Research report EMSE 2005-500-011 - September 2005. Submitted (European Journal of Operational Research)*, 2005.

[DoMUoK06] Department of Optimization ITWM Kaiserslautern Department of Mathematics University of Kaiserslauter. Library of location algorithms. *http://www.mathematik.uni-kl.de/lola/*, 2006.

[EG02] M. Ehrgott and X. Gandibleux. *Multiobjective Combinatorial Optimization - theory, methodology, and applications.* Kluwer Academic Publishers, Boston, pages 369-444, 2002.

[Ehr05a] M. Ehrgott. *Multicriteria Optimization.* Springer, pages 761-795, 2005.

[Ehr05b] M. Ehrgott. *Multicriteria Optimization.* Second Edition Springer, 2005.

[Erl78] Erlenkotter. A dual-based procedure for uncapacitated facility location, operations research. *Operations Research*, pages 992–1009, 1978.

[ERM89] P.K. Eswaran, A. Ravindran, and H. Moskowitz. Algorithms for nonlinear integer bicriterion problems. *Journal of Optimization Theory and Applications, 63(2)*, pages 261–279, 1989.

[FIS05] S. Favuzza, M.G. Ippolito, and E. Riva Sanseverino. Crowded comparison operators for constraints handling in nsga-ii for optimal design of the compensation system in electrical distribution networks. *Advanced Engineering Informatics 20*, pages 201–211, 2005.

[FP03] E. Fernandez and J. Puerto. Multiobjective solution of the uncapacitated plant location problem. *European Journal of Operational Research, Vol 145*, pages 483–715, 2003.

[Gal93a] R. Galvao. The use of lagrangean relaxation in the solution of uncapacitated facility location problems. *Location Science, 1*, pages 57–79, 1993.

[Gal93b] Galvo. The use of lagrangean relaxation in the solution of uncapacitated facility location problems. *Location Science*, pages 57–79, 1993.

[Geo74] Geoffrion. *Lagrangian Relaxation for Integer Programming.* Mathematical Programming Study, pages 82-114, 1974.

[GLM03] F. Glover, M. Laguna, and R. Marti. Library of location algorithms. *Springer-Verlag New York*, pages 519–537, 2003.

[Glo77] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences 8*, pages 156–166, 1977.

[GMJM02] F. Garcia, B. Melian, and J.M. Moreno-Vega J.A. Moreno. Scatter search for. multiple objective p-facility location problems. *Groupe de Travail ROADEF sur la Programmation Mathématique MultiObjectif, 6ème journée de travail*, 2002.

[GMK04] X. Gandibleux, H. Morita, and N. Katoh. A population-based metaheuristic for solving assignment problems with two objectives. *Journal of Mathematical Modelling and Algorithms*, 2004.

[GR87] D.E. Goldberg and J. Richardson. Adaptative niching via coevolutionary sharing. *Genetic algorithms and Their applications: Proceedings of the Second International Conference on Genetic algorithms*, pages 21–38, 1987.

[Hal76] J. Halpern. The location of a centdian convex combination on an undirected tree. *Journal of Regional Science, 16*, pages 237–245, 1976.

[Hal80] J. Halpern. Duality in the cent-dian of a graph. *Operations Research, Vol. 28*, pages 722–735, 1980.

[HD03] W.H. Horst and Z. Drezner. *Facility Location: Applications and Theory, 2^{nd} edition.* Springer, 2003.

[HJJK96] R. Houck, A. Jeffrey, Joines, and M. Kay. Comparison of genetic algorithms, random restart and two-opt switching for solving large location-allocation problems. *Department of Industrial Engineering, North Carolina State University, Raleigh, NC 27695-7906, U.S.A.*, 1996.

[HM97] P. Hansen and N. Mladenovic. Variable neighborhood search for the p-median. *Location Science, 5*, pages 207–226, 1997.

[HN93] J. Horn and N. Nafpliotis. Multiobjective optimisation using the niched pareto genetic algorithm. *Illinois Genetic Algorithms Laboratory (IlliGAL), report no. 93005*, 1993.

[HNG94] J. Horn, N. Nafpliotis, and D. Goldberg. A niched pareto genetic algorithm for multiple-objective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pages 82–87, 1994.

[KD04] A. Klose and A. Drexl. Facility location models for destination system design. *European Journal of Operational Research, Vol. 162*, pages 4–29, 2004.

[KH63] A.A. Kuehn and M.J. Hamburger. A heuristic program for locating warehouses. *Management Science 9*, pages 643–666, 1963.

[LJ82] R.F. Love and H. Juel. Properties and solution methods for large location-allocation problems. *Journal of Operation Research 33*, pages 443–452, 1982.

[LM03] M. Laguna and R. Mart. *Scatter Search: Methodology and Implementations in C.* Kluwer Academic Publishers,Boston, 2003.

[LV92] J-H. Lin and J. Vitter. ϵ-approximations with minimum packing constraint violation. *In Proceedings of the 24th Annual ACM Symposium on Theory of Computing,*, pages 771–782, 1992.

[Mar64] F.E. Maranzana. On the location of supply points to minimize transport costs. *Operational Research Quarterly*, pages 261–270, 1964.

[MHM95] E. Melachrinoudis and X. Wu H. Min. A multiobjective model for the dynamic location of landfills. *Location Science, Vol. 3*, pages 143–166, 1995.

[Mic98] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer; 3 edition, 1998.

[MLH00] N. Mladenovic, M. Labbé, and P. Hansen. Solving the p-center problem with tabu search and variable neighborhood search. *Computers and Operation Research Res. 24*, pages 1097–1100, 2000.

[MM00] E. Melachrinoudis and H. Minb. The dynamic relocation and phase-out of a hybrid, two-echelon plant/warehousing facility: A multiple objective approach. *European Journal of Operational Research, Vol 123*, pages 1–15, 2000.

[MV95] Muhlenbein and Voig. Gene pool recombination for the breeder genetic algorithm. *Proceedings of the Metaheuristics International Conference, Breckenridge, Colorado*, pages 19–25, 1995.

[MW94] K.E. Mathias and D. Whitley. Initial performance comparisons for the delta coding algorithm. In *International Conference on Evolutionary Computation*, pages 433–438, 1994.

[NP05] S. Nickel and J. Puerto. *Location Theory, A unified Theory*. Springer, 2005.

[NPRC05] S. Nickel, J. Puerto, and A.M. Rodriguez-Chia. *MCDM Location Problems*. Springer Verlag, pages 761-798, 2005.

[Per83] J. Perl. A unified warehouse location-routing analysis. *Ph.D. Dissertation, Department of Civil Engineering, Northwestern University, Evanston, IL*, 1983.

[PJ98] H. Pirkul and V. Jayaraman. A multi-commodity, multi-plant, capacitated facility location problem: formulation and efficient heuristic solution. *Computers & Operations Research 25*, pages 869–878, 1998.

[RB81] H.D. Sherali R. Bazaara. On the choice of step sizes in subgradient optimization. *European Journal of Operation Research, 7*, pages 380–388, 1981.

[REBY06] R.Galvo, L. Acosta Espejo, B. Boffey, and D. Yates. Load balancing and capacity constraints in a hierarchical location mode. *European Journal of Operational Research, Volume 172, Issue 2*, pages 631–646, 2006.

[RR96] K.E. Rosing and C. ReVelle. Heuristic concentration: two stage solution. construction. *European Journal of Operational Research, 97*, 1996.

[RRR+99] K.E. Rosing, C. Revelle, E. Rolland, D. Schilling, and J. Current. Heuristic concentration and tabu search: a head to head comparison. *European Journal of Operational Research 104*, pages 93–99, 1999.

[SCF06] C.G. Da Silva, J. Climaco, and J. Figueira. A scatter-search method for bi-criteria 0,1-knapsack problem. *European Journal of Operational Research 169*, pages 373–391, 2006.

[SD93] N. Srinavas and K. Deb. Multiobjective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation Journal 2(3)*, pages 221–248, 1993.

[Sin05] A. Sinha. Location, location, location and location. *A dissertation submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Algorithms, Combinatorics and Optimization, at the Tepper School of Business, Carnegie Mellon University*, 2005.

[SJ91] W.M. Spears and K.A. De Jong. *On the Virtues of Parametrized Uniform Crossover*. Spears, pages 230-236, 1991.

[Ste82] R.E. Steuer. On sampling the efficient set using weighted tchebycheff metrics. *Proceedings of the task force meeting on multiobjective and stochastic optimization*, pages 335–35, 1982.

[Sys89] G. Syswerda. Uniform crossover in genetic algorithms. *In H. Schaffer, editor, 3rd Int. Conf. on Genetic Algorithms*, pages 2–9, 1989.

[Tah87] H.A. Taha. *Operations Research: An introduction, 4th edition*. Collier Macmillan, London, 1987.

[TP68] M.B. Teitz and P.Bart. Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research*, pages 955–961, 1968.

[Whi83] R. Whitaker. A fast algorithm for the greedy interchange for large-scale clustering and median location problems. *INFOR*, 1983.

[Whi02] D. Whitley. *Delta Coding: An Iterative Search Strategy for Genetic Algorithms*. 2002.

[YIG06] M. Yagiura, T. Ibaraki, and F. Glover. A path relinking approach with ejection chains for the generalized assignment problem. *European Journal of Operational Research 169*, pages 548–569, 2006.

[Zit98] E. Zitzler. A tutorial on evolutionary multiobjective optimization. *Swiss Federal Institute of Technology (ETH), Zurich*, 1998.

[Zit99] E. Zitler. Evolutionary algorithms for multiobjective optimization: methods and applications. *Ph. D. Thesis, Swiss Federal Institute of Technology ETH*, 1999.

[Zit00] E. Zitler. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionay Computation Journal*, pages 173–195, 2000.

[Zit01] E. Zitler. Evolutionary multi-criterion optimization. *Lecture Notes in Computer Science*, 2001.

[Zit02] E. Zitzler. Performance assessment of multiobjective optimizers : an analysis and review, technical report 139. *Computer Engineering and Networks Laboratory (TIK), CH-8092, Switzerland*, 2002.

[ZT99] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Computers*, pages 257–271, 1999.