POLYTECH.MONS
beyond boundaries of learning

New metaheuristics for solving MOCO problems: application to the knapsack problem, the traveling salesman problem and IMRT optimization

Thibaut Lust

Aspirant FNRS

Dissertation accepted in fulfillment of the requirements
for the Belgian Doctorate Degree in Applied Sciences
and for the European Doctorate Degree

Jury Members:

Prof. A. Jaszkiewicz	Politechnika Poznańska (Poland)
Prof. P. Manneback	Faculté Polytechnique de Mons, President
Prof. M. Pirlot	Faculté Polytechnique de Mons, Secretary
Prof. T. Stützle	Université Libre de Bruxelles
Prof. E-G. Talbi	Université des Sciences et Technologies de Lille (France)
Prof. J. Teghem	Faculté Polytechnique de Mons, Supervisor
Prof. D. Tuyttens	Faculté Polytechnique de Mons, Co-supervisor
Prof. P. Lybaert	Faculté Polytechnique de Mons, Dean

2009-2010

European Doctorate Title

This dissertation has been presented to the Graduate Committee of the Faculté Polytechnique de Mons, Belgium, in fulfillment of the requirements for the European Doctorate Degree.

The European Doctorate Title has been defined and adopted by the European Community (EC) in April 1991, and it can be awarded when four conditions have been met:

1. The defense of the thesis has to be authorized by at least two professors from universities of two other EC countries.

   This authorization has been given by Prof. A. Jaszkiewicz (Politechnika Poznańska, Poland) and Prof. E-G. Talbi (Université des Sciences et Technologies de Lille, France).

2. At least one member of the Ph.D Committee is a professor from a university of another EC country.

   Prof. A. Jaszkiewicz and Prof. E-G. Talbi are members of the Ph.D Committee.

3. The thesis must be defended in another language of the EC than a national language of the country where it is presented.

   English is the language used for the defense of the thesis.

4. The work presented in the thesis must have been partly carried out in another EC country during a stay of at least three months.

   Part of the work reported in this thesis has been carried out while the author was a guest at the Institute of Computing Science of the Politechnika Poznańska, Poland where he had the opportunity to stay three months (October 2007, November 2008 and May 2009).

Acknowledgments

It is a great pleasure for me to thank those who made this thesis possible.

Foremost, I am heartily thankful to my supervisor, Prof. Jacques Teghem, who was abundantly helpful and offered invaluable support, guidance, assistance and friendship in numerous ways. This thesis would not have been possible without the trust he placed in me and his constant and remarkable enthusiasm.

I am very grateful to Prof. Marc Pirlot, who allowed me, as head of the laboratory of mathematics and operational research, to work in a friendly working environment. I would like also to thank him for his interest and constant stimulation.

It is impossible for me not to mention Prof. Andrzej Jaszkiewicz for hosting me during three months in his team in Poznań, and for showing me Poland and its different and delicious culinary specialties. The experience I had in Poland was very rewarding. I would like also to show him my gratitude for the work we have done together and for his valuable feedback and insights. I am also in debt with Prof. Roman Slowinski, as head of the laboratory of intelligent decision support systems, for having given me the opportunity to present my works during my different stays.

I would like to gratefully acknowledge Prof. Daniel Tuyttens for his constant interest and for the discussions that we had together which allowed to considerably improve the quality of this work.

I am deeply indebted to Prof. Thomas Stuëtlze who followed the progress of this thesis with a lot of attention and whose advices were always very helpful.

I would like to thank Prof. Pierre Manneback and Prof. El-ghazali Talbi for accepting to be members of the jury.

I had the chance to meet during conferences or summer schools, people who showed interest in my work, and I am especially deeply thankful to Prof. Matthias Ehrgott, Prof. Xavier Gandibleux, Prof. Luis Paquete and Dr. Anthony Przybyslki for the discussions and the help that they provided me.

I cannot forget to thank Céline Engelbeen from the Université Libre de Bruxelles, for introducing me the radiotherapy optimization problem. It was always a pleasure to work with her on this problem.

I thank my colleagues from the computer engineering and business management department for their support and friendship.

Thanks to all my friends, for their patience and affection.

I am very grateful to my family, and specially to my parents for their constant help and support.

Finally, I am deeply indebted to the FNRS for the financial support during the accomplishment of this thesis.

Summary

The motivation of this thesis is to develop new heuristic methods, based on metaheuristics, to solve multiobjective combinatorial optimization problems (MOCO problems). In MOCO problems, an optimal solution does not exist since conflicting objectives are considered to evaluate the solutions. On the other hand, a set of good compromise solutions (called efficient solutions) can be determined. An efficient solution is such that it is impossible to find another solution better on all objectives. The goal of multiobjective optimization is to provide to the decision maker the set of efficient solutions. The decision maker is then free to choose among all solutions proposed the one that corresponds the best to his preferences. The difficulties related to MOCO problems are the following: $\mathcal{NP}$-Hardness even when the single-objective counterpart is in $\mathcal{P}$, an exponential number of efficient solutions, and the presence of non-supported efficient solutions, harder to compute than supported ones. Accordingly, only heuristics methods based on metaheuristics have been developed in this thesis. Metaheuristics are general concepts including intensification and diversification techniques to explore effectively the search space and allowing to design a heuristic for a specific optimization problem, in order to obtain approximations of good quality in a reasonable computation time. The metaheuristics, being initially developed in a single-objective context, have to be adapted to tackle multiobjective problems, by integrating different mechanisms suited to the multiobjective nature of the problems.

Three new solving methods based on metaheuristics have been developed. The first one is a simple adaptation to multiobjective optimization of tabu search, called PRTS, with a selection of the neighbors based on Pareto dominance relations. The second one is a memetic algorithm, called MEMOTS, making use of PRTS as local search. The particularity of this algorithm is the selection of parents based on the density information given by a hypergrid built in the objective space, while most of the other memetic algorithms use scalarizing functions to pick out the parents. The last method, the two-phase Pareto local search (2PPLS), is an extension of the Pareto local search method, where in the place of one single initial solution, a population of solutions of good quality is used as starting point.

We have first applied the three methods to the multiobjective multidimensional knapsack problem. For this problem, the hybridization between MEMOTS and 2PPLS, where MEMOTS is used as a subroutine in 2PPLS to explore efficiently the neighborhood of a solution, gives better values for the indicators considered in this work, in lower computational times, than state-of-the-art results.

We have then considered the resolution of the biobjective traveling salesman problem. We have obtained with 2PPLS better results for many different types of instances than state-of-the-art results, for the indicators considered. We have also presented how to solve biobjective instances of a size never tackled before, that is 750 and 1000 cities. To this end, different speed-up techniques based on dominance relations performed in the data or based on the solutions previously generated, have been developed. These speed-up techniques are very efficient since they allow to greatly reduce the running time of 2PPLS without deteriorating the quality of the results.

The last part of this work concerns a very specific multiobjective problem that occurs in the treatment of cancer with intensity-modulated radiation therapy (IMRT). The IMRT is an advanced type of high-precision radiation allowing to optimally conforms radiation beams to tumor shapes. The MOCO problem related to this process is the decomposition of a positive integer matrix into a linear combination of binary matrices having to respect the consecutive ones property. The binary matrices allow to modelize the leaves of a multileaf collimator, that are used to block the radiations in certain locations. We have applied the 2PPLS method, in order to generate a good approximation of the efficient solutions of this problem. We have obtained promising results for random instances and real instances as well.

Résumé

La motivation de cette thèse est de développer de nouvelles méthodes heuristiques, basées sur les métaheuristiques, pour résoudre des problèmes d'optimisation combinatoire multiobjectifs (problèmes MOCO). Dans les problèmes MOCO, une solution optimale n'existe pas étant donné que des critères conflictuels sont considérés pour évaluer les solutions. Par contre, un ensemble de solutions "compromis" (appelées solutions efficaces) peut être déterminé. Une solution efficace est telle qu'il est impossible de trouver une autre solution meilleure sur tous les critères. Le but de l'optimisation multiobjectif est de fournir au décideur l'ensemble des solutions efficaces. Le décideur est ensuite libre de choisir parmi toutes les solutions proposées, la solution qui correspond le mieux à ses préférences. Les difficultés liées aux problèmes MOCO sont les suivantes : $\mathcal{NP}$-dureté même si le problème uniobjectif associé est dans $\mathcal{P}$, un nombre exponentiel de solutions efficaces, et la présence de solutions efficaces non supportées, plus difficile à générer que les solutions efficaces supportées. En conséquence, uniquement des méthodes heuristiques basées sur des métaheuristiques ont été développées dans cette thèse. Les métaheuristiques sont des concepts généraux incluant des techniques d'intensification et de diversification dans le but d'explorer de manière efficace l'espace de recherche et permettant de concevoir une heuristique pour un problème d'optimisation spécifique, afin d'obtenir des approximations de bonne qualité en un temps de calcul acceptable. Les métaheuristiques, étant à l'origine développées dans un contexte uniobjectif, devront être adaptées dans le but de traiter des problèmes multiobjectifs, par intégration de différents mécanismes adaptés à la nature multiobjectif des problèmes.

Trois nouvelles méthodes de résolution basées sur les métaheuristiques ont été développées. La première méthode est une simple adaptation à l'optimisation multiobjectif de la recherche tabou, appelée PRTS, qui comporte une sélection des voisins basée sur les relations de dominance de Pareto. La seconde méthode est un algorithme mémétique, appelé MEMOTS, utilisant PRTS comme recherche locale. La particularité de cet algorithme est la sélection des parents basée sur l'information de densité donnée par une hypergrille construite dans l'espace des objectifs, tandis que la plupart des autres algorithmes mémétiques utilisent des fonctions scalarisantes pour sélectionner les parents. La dernière méthode, la recherche locale Pareto à deux phases (2PPLS), est une extension de la recherche locale Pareto, où une population initiale de solutions de bonne qualité est utilisée à la place d'une solution initiale unique.

Nous avons tout d'abord appliqué les trois méthodes au problème du sac à dos multidimensionnel multiobjectif. Pour ce problème, l'hybridation entre MEMOTS et 2PPLS, où MEMOTS est utilisé comme sous-routine dans 2PPLS pour explorer efficacement le voisinage d'une solution, donne de meilleures valeurs pour les indicateurs considérés dans ce travail, dans des temps de calcul plus faibles, que les résultats issus de l'état de l'art.

Nous avons ensuite considéré la résolution du problème de voyageur de commerce biobjectif. Nous avons obtenu avec 2PPLS de meilleurs résultats pour de nombreux types d'instances que les résultats issus de l'état de l'art, pour les indicateurs considérés. Nous avons aussi présenté comment résoudre des instances biobjectifs d'une taille jamais traitée auparavant, c'est-à-dire 750 et 1000 villes. Dans ce but, différentes techniques d'accélération basées sur des relations de dominance sur les données ou basées sur les solutions préalablement générées, ont été développées. Ces techniques d'accélération sont très efficaces étant donné qu'elles permettent de réduire fortement le temps de calcul de 2PPLS sans détériorer la qualité des résultats.

La dernière part de ce travail concerne un problème multiobjectif très spécifique qui intervient dans le traitement du cancer avec la radiothérapie conformationnelle avec modulation d'intensité (RCMI). La RCMI est un type de radiothérapie avancé, permettant d'adapter de manière optimale les rayons des radiations aux formes des tumeurs. Le problème MOCO associé à ce procédé est la décomposition de matrices entières positives en une combinaison linéaire de matrices binaires devant respecter la propriété des uns consécutifs. Les matrices binaires permettent de modéliser les lames d'un collimateur multilames, qui sont utilisées pour bloquer les radiations dans certains endroits. Nous avons appliqué la méthode 2PPLS, dans le but de générer une bonne approximation des solutions efficaces de ce problème. Nous avons obtenu des résultats prometteurs pour des instances aussi bien aléatoires que réelles.

Contents

Abbreviations

- 2PPLS: Two-Phase Pareto Local Search
- ACO: Ant Colony Optimization
- bTSP : biobjective Traveling Salesman Problem
- CENSGA: Controlled Elitist Non-dominated Sorting Genetic Algorithm
- CO: Combinatorial Optimization
- DPR: Double Pareto Ranking
- DPX: Distance Preserving crossover
- EA: Evolutionary Algorithm
- EDA: Estimation of Distribution Algorithm
- FPTAS: Fully Polynomial Time Approximation Scheme
- GA: Genetic Algorithm
- GRASP: Greedy Randomized Adaptive Search Procedure
- IMMOGLS: Ishibuchi and Murata Multiple Objective Genetic Local search
- IMRT: Intensity-Modulated Radiation Therapy
- KP: Knapsack Problem
- LK: Lin-Kernighan
- LS: Local Search
- MA: Memetic Algorithm
- MEMOTS: MEmetic MultiObjective Tabu Search
- MKP: Multidimensional Knapsack Problem
- MLC: MultiLeaf Collimator
- MO: MultiObjective
- MOACO: MultiObjective Ant Colony Optimization
- MOCO: MultiObjective Combinatorial Optimization
- MOGA: MultiObjective Genetic Algorithm
- MOGLS: MultiObjective Genetic Local Search
- MOILP: MultiObjective Integer Linear Programming
- MOKP: MultiObjective Knapsack Problem
- MOLP: MultiObjective Linear Programming
- MOLS: MultiObjective Local Search
- MOMA: MultiObjective Memetic Algorithm
- MOMH: MultiObjective MetaHeuristic

- MOMILP: MultiObjective Mixed Integer Linear Programming
- MOMKP: MultiObjective Multidimensional Knapsack Problem
- MONLP: MultiObjective Non-Linear Programming
- MOO: MultiObjective Optimization
- MOP: MultiObjective Problem
- MOSA: MultiObjective Simulated Annealing
- MOTGA: MultiObjective Tchebycheff Genetic Algorithm
- MOTS: MultiObjective Tabu Search
- MOTSP: MultiObjective Traveling Salesman Problem
- M-PAES: Memetic Pareto Archived Evolution Strategy
- NSGA : Non-dominated Sorting Genetic Algorithm
- PAES: Pareto Archived Evolution Strategy
- PCGA: Pareto Converging Genetic Algorithm
- PLS: Pareto Local Search
- PMA: Pareto Memetic Algorithm
- PRTS: Pareto Ranking Tabu Search
- PSA: Pareto Simulated Annealing
- PTAS: Polynomial Time Approximation Scheme
- REMO: Restricted Evolutionary Multiobjective Optimizer
- SA: Simulated Annealing
- SAC: Scalarized Acceptance Criterion
- SMOSA: Serafini's MultiObjective Simulated Annealing
- SPEA: Strength Pareto Evolutionary Algorithm
- SS: Scatter Search
- TS: Tabu Search
- TSP: Traveling Salesman Problem
- ZMKP: the instances of Zitzler of the Multidimensional Knapsack Problem

1 Introduction

Let's imagine this situation:

We are in 2022. In 100 years, the heat on the earth will be so high that no special jumpsuit will be strong enough to protect you from the UV rays emitted by the sun. It is high time to find a new planet where the mankind will be able to live.

Let's think up that you work as an engineer in NASA. Today, your mission is to explore the space on board of a spaceship. You have to discover new planets that are livable, that is with an atmosphere constituted of at least 15% of oxygen and no more than 25% with 20% the ideal value, a mean temperature included between -20°C and 35°C with 18°C the ideal value and constitued of a proportion of water between 20% and 80% with 60% the ideal value. Unfortunately, scientists have proved that in this space, the requirements are conflicting and it will be impossible to find the ideal planet, that is with an atmosphere composed of 20% of oxygen, a mean temperature of 18°C and a proportion of water of 60%.

In order to find an interesting planet, you enter in your ship, take off and rapidly hit the mark: a black hole which is the gate of a hyper-dimensional space full of new planets. You put your special glasses that allow to see in this hyper-dimensional space and are now ready to explore each planet.

But as shown in Figure 1.1, planets are everywhere. You are thus confronted with many decisions to take: which paths to take to explore the planets, in which order, which planets to retain once they have been visited, and the more crucial point: which planets to visit given that for fuel capacity you have only two days to explore this hyper-dimensional space. It will be thus impossible to explore all planets, but only a small proportion, since the number of planets present in this space is huge.

What do you need for? If you randomly visit the planets, you will only explore a sample not very well-chosen. A device that gives indications for finding interesting planets would help a lot. But what is an interesting planet? We can define an interesting planet as follows. First of all, an interesting planet is a planet that respects the requirements defined above. But that is not all, since a planet will be interesting and *efficient* only if for this planet, we cannot find a better planet on all requirements (that is closer for all requirements to the ideal planet). Hence, if you want to be sure that this planet is efficient, you have to visit all planets, to be able to certify that we cannot find a planet better on all requirements. But as there is not enough time to explore all the space, the planet can only be *potentially* efficient. It means that, during your search, you have not found a better planet on all requirements.

The device, kind of GPS, should give you the interesting regions to explore in this hyper-dimensional space and also, once this interesting region has been defined, how to go from one planet to another.

After this two-day trip, you will give to the manager of NASA, the set of potentially efficient planets that you have obtained. Note that during your trip, you constantly update the set, since a potentially efficient planet cannot be efficient anymore if you find a planet better on all requirements.

Your mission is then finished and the manager of NASA, with the help of other heads of the world, will be free to choose the most convenient planet from the list of potentially efficient planets.

Figure 1.1: You enter in this hyper-dimensional space: where to go and which planets to explore? (source: Wassily Kandinsky, Several Circles, 1926).

This is what this thesis is all about: finding efficient solutions in a hyper-dimensional space. The hyper-dimensional space is defined by the structure of a combinatorial problem. For these problems, it is easy to find a feasible solution, but it is hard to show that the solution is efficient. The problems are multiobjective (MO) (or multicriteria), that is, each solution is evaluated according to different objectives (or criteria). The aim is to develop methods, based on metaheuristics, with the aim of finding a good approximation of the efficient solutions. The running time of the method will be limited to a reasonable value.

In Table 1.1, we can see the comparisons between the terms of the NASA problem and the terms used for multiobjective combinatorial optimization (MOCO) problems.

Table 1.1: Analogy NASA problem - MOCO problem

NASA problem	**MOCO problem**
Planet	Solution
Interesting planet	Feasible solution
Efficient planet	Efficient solution
Hyper-dimensional space (only visible with special glasses)	Hyper-dimensional space defined by the combinatorial structure of the problem (impossible to represent easily)
Device, kind of GPS	Method based on metaheuristics
Two days to explore the space	Limited running time of the method

Actually, to solve a MOCO problem, three different approaches can be followed. In the *a posteriori* approach, all the efficient solutions are first generated, that is, the solutions that cannot be improved simultaneously on all criteria. It is not possible to improve the value of one of the criteria of these solutions without deteriorating at least one other criterion. A method to generate all efficient solutions can simply be to enumerate all feasible solutions (since the number of feasible solutions is finite) and to retain the efficient ones. Once this task has been realized, the decision maker is free to choose among all solutions the one that corresponds the best to his preferences. For this specific task, since the number of efficient solutions can be very large, different methods coming from the multicriteria decision making field can help [239].

Another possibility, called *a priori* approach, is to first ask the decision maker what are his preferences among all criteria, that is what are his priorities, in order to focus the search to solutions that correspond to his criteria. In the end, only one solution, or a very small set of solutions, is proposed to the decision maker.

A last possibility is to *interact* with the decision maker along the process of generating solutions. In this *interactive* approach, we ask the decision maker, for instance, to establish his preferences among different (partial) solutions, in order to guide the search, and to finally obtain a solution that suits him.

In this thesis, we only consider the first possibility, that is, we assume that nothing is known about the preferences of the decision maker. Therefore, the only way to tackle multiobjective problems is to generate all the efficient solutions, since it is impossible to give a preference among these solutions. Once this task has been done, one solution among the set of efficient solutions has to be chosen, by using various methods coming from the multicriteria decision making field [239]. But this step is out of the scope of the thesis.

As said above, one possibility to generate all efficient solutions is to simply enumerate all feasible solutions and to retain the efficient ones (this is a point that differentiates the combinatorial optimization problems from continuous optimization problems, where the number of feasible solutions is infinite). But in this work, we will only tackle combinatorial optimization problems for which it is impossible, in a reasonable time, to enumerate all feasible solutions (the number of feasible solutions is exponential in the size of the instance). Moreover, for many of

the problems treated, it will be impossible to use an elaborated exact method to generate all the efficient solutions, given the size and the complexity of the problems considered. Therefore, the thesis relates to heuristic methods, that is methods that allow to approximate the set of efficient solutions, in a reasonable time, but without guaranteeing that all efficient solutions have been generated and even that all solutions in the approximation are indeed efficient. That is why the solutions produced by a heuristic are often called potentially efficient solutions. The characterization of how good an approximation is will be an important question discussed in this work.

The heuristics developed will be based on metaheuristics, that is general concepts including intensification and diversification techniques to explore the search space and allowing to design a heuristic for a specific optimization problem. The metaheuristics, being initially developed in a single-objective context, will have to be adapted to tackle multiobjective problems. We will see that different mechanisms suited to the multicriteria nature of the problems will have to be developed.

Combinatorial optimization problems are hard to solve, multicriteria combinatorial problems are even harder! But we will see that the single-objective version of combinatorial optimization problems considered in this work can be of a very good help to resolve the multicriteria version, and that the heuristics developed can achieve results close to exact results.

1.1 Motivation

The motivation of this thesis is to develop new heuristic methods, based on metaheuristics, to solve MOCO problems, that is, to generate approximations of good quality to the efficient set.

We will see in an overview about the methods for solving MOCO problems that many metaheuristics have already been adapted to tackle MOCO problems. But most of the methods include many parameters and are sometimes so complex that it is difficult to deeply understand the behavior of these methods. It makes the application of these methods to MOCO problems hard and not necessary efficient. For the new methods developed in this thesis, two features are expected: simplicity and effectiveness. The methods should be as simple as possible to easily adapt them to different MOCO problems and to give better results than state-of-the-art results on different MOCO problems. We also intend to give through this work a better knowledge concerning the efficient solutions of MOCO problems, as well as introducing new techniques to solve new MOCO problems. Another motivation is to apply the methods developed to real MOCO problems.

1.2 Contributions

The different contributions are listed below:

- We propose three new multiobjective metaheuristics for the resolution of MOCO problems: the Pareto ranking tabu search (PRTS), the memetic multiobjective tabu search (MEMOTS) and the two-phase Pareto local search method (2PPLS).
- We apply the new methods to two classical MOCO problems: the multiobjective multidimensional knapsack problem (MOMKP) and the multiobjective traveling salesman problem (MOTSP).
- The methods and their parameters are analyzed in detail on these two problems.
- New tricks to tackle MOCO problems are introduced: the data perturbation technique, and speed-up techniques in order to solve large-scale MOCO problems.
- The results obtained with our methods are compared, through a large a set of numerical experiments, to state-of-the-art methods, by using different indicators and tools to assess the quality of the approximations obtained. With this aim, the notion of ideal set

is introduced. We have obtained new state-of-the-art results for the MOMKP and the MOTSP.

- We present the multiobjective version of a combinatorial problem that occurs in intensity-modulated radiotherapy treatment (IMRT). For this problem, we apply the two-phase Pareto local search method on different random instances and on real instances. This is the first multiobjective approach of this problem.

Most of the contributions have been published in international journals or in the proceedings of international conferences, and have also been presented orally in international conferences. We present below the main publications generated during this work:

- [162]: *MEMOTS: a memetic algorithm integrating tabu search for combinatorial multiobjective optimization*, T. Lust and J. Teghem, RAIRO Oper. Res. 42 (2008) 3-33.

  We present in this paper a new multiobjective memetic algorithm method called MEMOTS. In current multiobjective memetic algorithms, the parents used for recombination are randomly selected. We improve this approach by using a dynamic hypergrid, which allows to select a parent located in a region of minimal density. The second parent selected is a solution close, in the objective space, to the first parent. A local search is then applied to the offspring. The local search used in MEMOTS is the new multiobjective tabu search, called PRTS. We show on the multidimensional multiobjective knapsack problem that if the number of objectives increases, it is preferable to have a diversified search rather than using an advanced local search. We compare the memetic algorithm MEMOTS to other multiobjective memetic algorithms by using different quality indicators and we show that MEMOTS obtains better values for most of the indicators considered.

- [165]: *Two-phase Pareto local search for the biobjective traveling salesman problem*, T. Lust and J. Teghem, Journal of Heuristics, accepted the 5th February 2009, in press.

  In this work, we present a method, called two-phase Pareto local search (2PPLS), to find a good approximation of the efficient set of the biobjective traveling salesman problem. In the first phase of the method, an initial population composed of a good approximation of the extreme supported efficient solutions is generated. We use as second phase a Pareto local search method applied to each solution of the initial population. We show that using the combination of these two techniques: good initial population generation plus Pareto local search gives better results than state-of-the-art algorithms. Two other points are introduced: the notion of ideal set and a simple way to produce near-efficient solutions of multiobjective problems, by using an efficient single-objective solver with a data perturbation technique.

- [159]: *Speed-up techniques for solving large-scale biobjective TSP*, T. Lust and A. Jaszkiewicz, Computers & Operations Research 37 (2010) 521-533.

  In this paper, we present the two-phase Pareto local search (2PPLS) method with speed-up techniques for the heuristic resolution of the biobjective traveling salesman problem. 2PPLS is a state-of-the-art method for this problem. However, because of its running time that strongly grows with the instances size, the method can be hardly applied to instances with more than 200 cities. We thus adapt some speed-up techniques used in single-objective optimization to the biobjective case. The proposed method is able to solve instances with up to 1000 cities in a reasonable time with no, or very small, reduction of the quality of the generated approximations.

- [164]: *The multiobjective traveling salesman problem: a survey and a new approach*, T. Lust and J. Teghem, accepted in March 2009 for publication as chapter for a Springer book on advances in multi-objective nature inspired computing (C. Coello Coello, C. Dhaenens and L. Jourdan, editors), in press.

The traveling salesman problem (TSP) is one of the more challenging problems of combinatorial optimization. In this paper we consider the multiobjective TSP for which the aim is to obtain or to approximate the set of efficient solutions. In a first step, we classify and describe briefly the existing works, essentially based on the use of metaheuristics. In a second step, we summarize the results obtained with the two-phase Pareto local search.

- [163]: *Multiobjective decomposition of positive integer matrix: application to radiotherapy*, T. Lust and J. Teghem, in the proceedings of the 5th EMO conference in Nantes, April 2009, 335-349.

  We consider the following problem: to decompose a positive integer matrix into a linear combination of binary matrices that respect the consecutive ones property. The positive integer matrix corresponds to fields giving the different radiation beams that a linear accelerator has to send throughout the body of a patient. Due to the inhomogeneous dose levels, leaves from a multileaf collimator are used between the accelerator and the body of the patient to block the radiations. The leaves positions can be represented by segments, that are binary matrices with the consecutive ones property. The aim is to find a decomposition that minimizes the irradiation time, and the setup-time to configure the multileaf collimator at each step of the decomposition. We propose for this $\mathcal{NP}$-Hard multiobjective problem a heuristic method, based on the two-phase Pareto local search method. Experimentations are carried out on different size instances and the results are reported.

1.3 Outline of the thesis

In the next chapter (chapter 2), we introduce the multiobjective optimization problems field and define the basic notions. We then focus the presentation on MOCO problems and we define the two traditional MOCO problems considered in this work: the multiobjective multidimensional knapsack problem (MOMKP) and the multiobjective traveling salesman problem (MOTSP). We expose then an overview of methods for solving MOCO problems where we mainly present multicriteria metaheuristics (MOMHs) (chapter 3). We also expose in this chapter how to assess the quality of an approximation in the multicriteria case. The two next chapters are devoted to a state-of-the-art of the two classical MOCO problems tackled in this work: the first deals with the MOMKP (chapter 4) while the second with the MOTSP (chapter 5). The following chapters present the contributions. In chapter 6, the new methods of resolution; PRTS, MEMOTS and 2PPLS, are exposed. In chapter 7, we apply the methods to the MOMKP. In chapter 8, we consider the biobjective TSP (bTSP) and the application of 2PPLS. Data perturbation and speed-up techniques are introduced in this chapter. Finally, in chapter 9, we present a MOCO problem that occurs in radiotherapy treatment. The problem consists in decomposing positive integer matrices, by considering three criteria to evaluate the decomposition.

1.4 Genesis of the thesis

Maybe it is interesting to say a word about the chronology of the results of this thesis, since the results are not necessarily exposed in the chronological order. Initially, it is a tabu search-based method that has been developed. As this method was not efficient enough to be competitive with other MOMHs, we have integrated it in a population-based method to thereby obtain the MEMOTS method. We have applied MEMOTS to the MOMKP and we have obtained good results. We then have tried to adapt MEMOTS to the MOTSP but it was not easy; many parameters had to be tuned and many possibilities were possible. Therefore, the 2PPLS method has been developed, which is composed of only two phases: an initial population generation and the application of a neighborhood. With this method, we have generated very good results for the bTSP in a very simple way. We have then integrated speed-up techniques to solve large-scale instances of the bTSP (this work has partially been realized during the different stays in

Poznań). After that, quite long work has been started on the radiotherapy application: for this problem, we started from nothing, so we read quite a lot and it is only after few months that the multiobjective problem has been defined and validated. The adaptation of a method to this problem was not simple, given the particularities of the problem as well as the few non-dominated points. It is first a branch and bound-based method that we tried but without success. Finally, it is following a tutorial in Sicily about the vehicle routing problem that the idea for the neighborhood arrived and allowed to obtain good results. Aside this problem, we were also working on the adaptation of 2PPLS to the MOMKP and we rapidly obtained good results, but it is only at the end of the thesis that the experiments and the exposition of the results have been carried out. These last results have not yet been published or even presented in conferences.

2 Multiobjective optimization

We start this chapter by giving a general formulation of multiobjective problems (MOPs), and among these problems, we present the multiobjective integer linear programming (MOILP) problems as well as the multiobjective combinatorial optimization (MOCO) problems. We then expose general concepts that share these problems: Pareto optimality, complete and minimal set of efficient solutions, ϵ-dominance, lexicographic optimality, ideal points and nadir points. We then present the different notions specific to MOCO problems, that is supported and non-supported efficient solutions, complexity, intractability and connectedness of efficient solutions. Two MOCO problems, tackled in this work, are then introduced: the multiobjective multi-dimensional knapsack problem (MOMKP) and the multiobjective traveling salesman problem (MOTSP).

2.1 Multiobjective optimization (MOO)

It is first the multiobjective problems (MOP) with continuous variables which caught the attention of the researchers.

We formulate a general MOP as follows:

$$\begin{aligned} &\text{“}\min_{x}\text{”}\, f(x) && \text{(MOP)} \\ &\text{subject to } g(x) \leq 0 \\ &\qquad x \in \mathbb{R}^n \end{aligned} \tag{2.1}$$

$$\begin{aligned} x \in \mathbb{R}^n &\rightarrow n \text{ variables}, i = 1, \ldots, n \\ f : \mathbb{R}^n \rightarrow \mathbb{R}^p &\rightarrow p \text{ objective functions}, k = 1 \ldots p \\ g : \mathbb{R}^n \rightarrow \mathbb{R}^m &\rightarrow m \text{ constraints}, j = 1 \ldots, m \end{aligned}$$

In this problem, p objective functions $(f_1, f_2, \ldots, f_p)$ have to be minimized (without loss of generality, we will consider minimization problems). A feasible solution x is a vector of n variables belonging to $\mathbb{R}$ and having to respect the m constraints g of the problem $(g_1, g_2, \ldots, g_m)$. Therefore, the feasible set in decision space is given by $\mathcal{X} = \{x \in \mathbb{R}^n : g(x) \leq 0\}$. The feasible set in objective space, that is the evaluation of the feasible set, is given by $\mathcal{Y} = f(\mathcal{X}) = \{f(x) : x \in \mathcal{X}\}$. The number p of objective is often limited to a small number, and only in the case of p equal to 2 or 3, it will be possible to easily represent $\mathcal{Y}$. In this work, the number of objectives will be equal to maximum 3. The case of $p > 3$ is known under the name many-objective optimization [72], and requires different techniques of resolution than the case of $p = 2$ or $p = 3$. We also make the assumption that it does not exist a feasible solution simultaneously minimizing each objective (the objectives are supposed to be conflicting). That is why the “min” term is put between quotation marks.

In this general formulation, the objective functions and constraints can be non-linear or linear (see the book of Steuer [214] for multiobjective *linear programming* (MOLP) problems

and of Miettinen [174] for multiobjective *non-linear programming* (MONLP) problems) and the variables are continuous. Therefore an infinite number of solutions are feasible.

We have represented in Figure 2.1 an example of feasible set in objective space for a general MOP, in the case of $p = 2$.

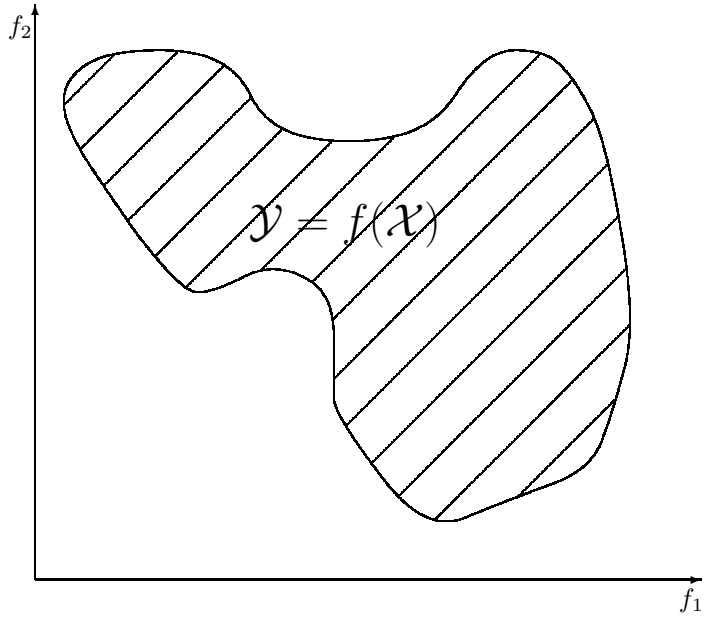

Figure 2.1: Example of feasible set in objective space of a general MOP: $\mathcal{Y} = f(\mathcal{X}) = \{f(x) : x \in X\}$, with $p = 2$.

However, it is well-known that discrete variables are often unavoidable in the modeling of many applications. Let us consider for instance a multiobjective *integer linear programming* (MOILP) problem of the form:

$$\begin{array}{rcll} \text{“}\min_x\text{”} f(x) & = & Cx & \text{(MOILP)} \\ \text{subject to } Ax & \leq & b & \\ x & \in & \mathbb{N}^n & \end{array} \tag{2.2}$$

$$\begin{array}{rcl} x \in \mathbb{N}^n & \longrightarrow & n \text{ variables}, i = 1, \ldots, n \\ C \in \mathbb{Z}^{p \times n} & \longrightarrow & p \text{ objective functions}, k = 1, \ldots, p \\ A \in \mathbb{Z}^{m \times n} \text{ and } b \in \mathbb{Z}^{m \times 1} & \longrightarrow & m \text{ constraints}, j = 1, \ldots, m \end{array}$$

Comparing to the preceding formulation, we remark that the constraints and the objectives are linear. But the main difference is that the feasible solutions x are vectors of n integer variables ($x \in \mathbb{N}^n$). As a consequence, the set in decision space, given by $\mathcal{X} = \{x \in \mathbb{N}^n : Ax \leq b\}$ is generally of finite cardinality. The feasible set in objective space is defined by $\mathcal{Y} = f(\mathcal{X}) = \{Cx : x \in \mathcal{X}\} \subset \mathbb{Z}^p$ and is generally also of finite cardinality.

We have represented in Figure 2.2 an example of feasible set in objective space, for a MOILP, in the case of $p = 2$, composed of 15 different points.

Already in the 80ies, several methods have been proposed to solve MOILP, see [221] for a survey.

Later, various multiobjective *combinatorial* optimization (MOCO) problems have been considered, that we formulate as follows:

$$\begin{array}{rcll} \text{“}\min_x\text{”} f(x) & = & Cx & \text{(MOCO)} \\ \text{subject to } Ax & \leq & b & \\ x & \in & \{0,1\}^n & \end{array} \tag{2.3}$$

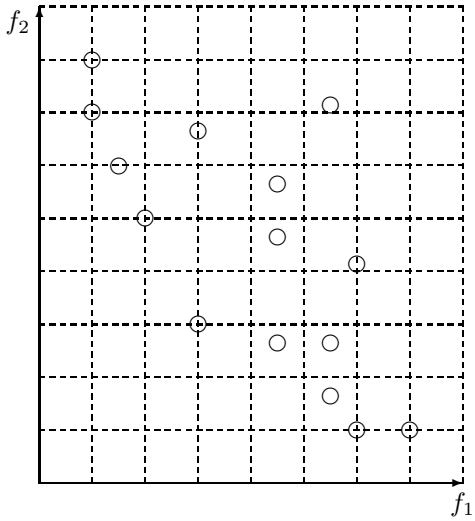

Figure 2.2: Example of feasible set in objective space of a MOILP: $\mathcal{Y} = f(\mathcal{X}) = \{f(x) : x \in \mathcal{X}\}$, with $p = 2$.

$$\begin{array}{rcl} x \in \{0,1\}^n & \longrightarrow & n \text{ variables}, i = 1, \ldots, n \\ C \in \mathbb{Z}^{p \times n} & \longrightarrow & p \text{ objective functions}, k = 1, \ldots, p \\ A \in \mathbb{Z}^{m \times n} \text{ and } b \in \mathbb{Z}^{m \times 1} & \longrightarrow & m \text{ constraints}, j = 1, \ldots, m \end{array}$$

A combinatorial structure is associated to this problem, which can be path, tree, flow, tour, etc., which distinguishes the combinatorial optimization (CO) field with the ILP field [245].

During the last 15 years, there has been a notable increase of the number of studies on MOCO problems. From the first survey [230] in 1994 till [56] in 2002, a lot of papers have been published and this flow is still increasing.

We present in the next subsections different concepts, applicable to all MOPs. However, as we only consider in this work MOCO problems, the case of these problems (a discrete feasible set $\mathcal{Y}$ in objective space) will only be considered in the following figures. We start by exposing how to characterize the interesting solutions that "minimize" the different objectives, through the notion of Pareto optimality.

2.1.1 Pareto optimality

In order to make the distinction between the different vectors of $\mathbb{R}^p$, we define three different orders.

Let y^1 and $y^2 \in \mathbb{R}^p$.

1. Weak component-wise order: $y^1 \preceq y^2 \Leftrightarrow y^1_k \leq y^2_k \ \ \forall k \in \{1, \ldots, p\}$. This relation is reflexive, transitive and antisymmetric.
2. Component-wise order: $y^1 \prec y^2 \Leftrightarrow y^1_k \leq y^2_k \ \ \forall k \in \{1, \ldots, p\}$ and $y^1 \neq y^2$ (for at least one k, $y^1_k < y^2_k$). This relation is irreflexive, transitive and asymmetric.
3. Strict component-wise order: $y^1 \prec\prec y^2 \Leftrightarrow y^1_k < y^2_k \ \ \forall k \in \{1, \ldots, p\}$. This relation is irreflexive, transitive and asymmetric.

These three orders allow to define three different dominance relations of Pareto for a MOP. Let x and x' two feasible solutions and respectively $f(x)$ and $f(x')$ their evaluations, objective vectors of p components.

Definition 1 *Weak dominance relation of Pareto: if $f(x) \preceq f(x')$, $f(x)$ weakly dominates $f(x')$.*

Definition 2 *Dominance relation of Pareto: if $f(x) \prec f(x')$, $f(x)$ dominates $f(x')$.*

Definition 3 *Strict dominance relation of Pareto: if $f(x) \prec\prec f(x')$, $f(x)$ strictly dominates $f(x')$.*

The strict dominance implies the dominance and the dominance implies the weak dominance. We have thus:

$$f(x) \prec\prec f(x') \Rightarrow f(x) \prec f(x') \Rightarrow f(x) \preceq f(x').$$

The orders $\preceq, \prec, \prec\prec$ are not total orders (that is reflexive, antisymmetric, transitive and connected orders [53]). That means that it is not always possible to compare the objective vectors.

For instance, in Figure 2.3, we have represented four vectors in the objective space, for $p = 2$. All pairs of vectors can be compared with the dominance relation of Pareto except the couple (b, c), since b is *incomparable* to c (denoted by $b||c$), that is $b \npreceq c$ and $c \npreceq b$.

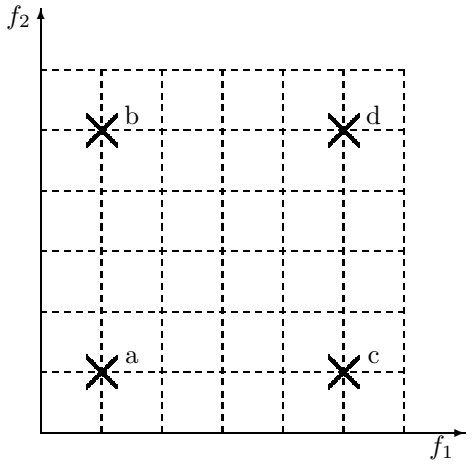

Figure 2.3: Illustration of the dominance relations: all pair of vectors can be compared with the dominance relation of Pareto ($a \prec\prec d$, $a \prec b$, $a \prec c$, $b \prec d$, $c \prec d$) except the couple (b, c).

Therefore, in MOO, the solutions cannot always be compared. This is the fundamental difference between single and multiobjective optimization.

We can now define an efficient solution, a non-dominated point, the efficient set and the Pareto front.

Definition 4 *Efficient solution: a feasible solution $x^* \in \mathcal{X}$ is called* efficient *if there does not exist any other feasible solution $x \in \mathcal{X}$ such that $f(x) \prec f(x^*)$.*

Definition 5 *Efficient set: the efficient set denoted by X_E contains all the efficient solutions.*

Definition 6 *Non-dominated point: the image $f(x^*)$ in objective space of an efficient solution x^* is called a non-dominated point.*

Definition 7 *Pareto front: the image of the efficient set in $\mathcal{Y}$, equal to $f(X_E)$, is called the Pareto front (or non-dominated frontier), and is denoted by Y_N.*

The notion of efficient solution is known under different denotations, for instance as Pareto optimal solution, but in this work, we will always use the term efficient solution.

We also add the definition of a weakly efficient solution, a weakly non-dominated point and the weakly efficient set:

Definition 8 *Weakly efficient solution: a feasible solution $x^+ \in \mathcal{X}$ is called* weakly efficient *if there does not exist any other feasible solution $x \in \mathcal{X}$ such that $f(x) \prec\prec f(x^+)$.*

Definition 9 *Weakly non-dominated point: the image $f(x^+)$ in objective space of a weakly efficient solution x^+ is called a weakly non-dominated point.*

Definition 10 *Weakly efficient set: the weakly efficient set denoted by X_{wE} contains all the weakly efficient solutions. The image of X_{wE} in $\mathcal{Y}$, equal to $f(X_{wE})$, is called Y_{wN}.*

We have represented in Figure 2.4 the image in objective space of the weakly efficient set and the Pareto front of the example of Figure 2.2.

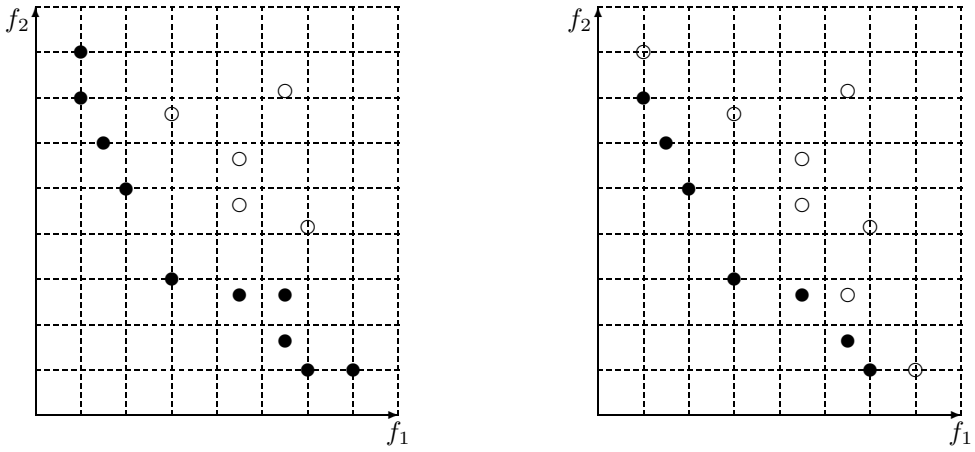

Figure 2.4: On the left: the ten filled points represent the image of the weakly efficient solutions in objective space (Y_{wN}). On the right, the seven filled points represent the Pareto front. Therefore, three points are weakly non-dominated points but are not non-dominated points.

As a weakly efficient solution can be dominated by an efficient solution, these solutions are not useful in practice, and we are only interested in the efficient set. Nevertheless, it is interesting to make the distinction for a theoretical point of view, as we will see later.

Another interesting equivalent definition to characterize an efficient solution is as follows. This definition is based on a geometric interpretation of the dominance relation of Pareto. Let us first define subsets of $\mathbb{R}^p$ as follows, based on the (weak, strict) component-wise orders:

- $\mathbb{R}^p_{\succeq} := \{y \in \mathbb{R}^p : y \succeq 0\}$, the nonnegative orthant of $\mathbb{R}^p$;
- $\mathbb{R}^p_{\succ} := \{y \in \mathbb{R}^p : y \succ 0\} = \mathbb{R}^p_{\succ} \backslash \{0\}$;
- $\mathbb{R}^p_{\succ\succ} := \{y \in \mathbb{R}^p : y \succ\succ 0\} = \text{int } \mathbb{R}^p_{\succeq}$ (interior of $\mathbb{R}^p_{\succeq}$), the positive orthant of $\mathbb{R}^p$.

Definition 11 *Efficient solution: a feasible solution $x^* \in \mathcal{X}$ is called* efficient *if $\mathcal{Y} \cap \left\{f(x^*) - \mathbb{R}^p_{\succeq}\right\} = \left\{f(x^*)\right\}$ where $\left\{f(x^*) - \mathbb{R}^p_{\succeq}\right\} = \left\{f(x^*) - y^p : y^p \in \mathbb{R}^p_{\succeq}\right\}$.*

This relation is illustrated in Figure 2.5. In this definition, for a point $f(x)$ we check if it exists another point $f(x')$ located to the left and below (in the direction of $-\mathbb{R}^p_{\succeq}$) that point. If not, the solution x is an efficient solution.

Even if other approaches exist to tackle MOP problems (aggregation of the objectives with a utility function, hierarchy of the objectives, goal programming, interactive method to generate a " good compromise ": see [220]), in this work we are only interested in the determination, or in the approximation of X_E and Z_N.

2.1.2 Complete and minimal set of efficient solutions

We introduce here a classification of the set X_E based on the notion of equivalent solutions [101].

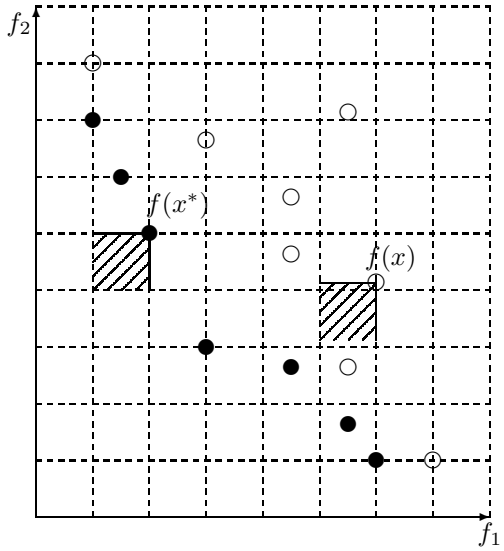

Figure 2.5: In this figure, we see that the solution x^* is efficient since $\mathcal{Y} \cap \left\{f(x^*) - \mathbb{R}^p_{\preceq}\right\} = \left\{f(x^*)\right\}$. On the other hand, the solution x is not efficient since $\mathcal{Y} \cap \left\{f(x) - \mathbb{R}^p_{\preceq}\right\} \neq \left\{f(x)\right\}$ and contains five points different from $f(x)$.

Definition 12 *Equivalent solutions: two solutions $x_1, x_2 \in X_E$ are equivalent if $f(x_1) = f(x_2)$.*

We define now a complete set and a minimal complete set.

Definition 13 *Complete set: a complete set X_{E_c} is a subset of X_E such that each $x \in \mathcal{X} \setminus X_{E_c}$ is weakly dominated by at least one $x \in X_{E_c}$, that is either dominated by or equivalent to at least one $x \in X_{E_c}$. In other words, for each non-dominated point $y \in Y_N$ there exists at least one $x \in X_{E_c}$ with $f(x) = y$.*

Definition 14 *Minimal complete set (or reduced efficient set): a minimal complete set X_{E_m} is a complete set without equivalent solutions. Any complete set contains at least a minimal complete set.*

The efficient set is unique since it contains all the efficient solutions (this set is also sometimes called *maximal* set) but the minimal set is generally not.

It is important to say that, in this work, we will only try to find an approximation of a minimal complete set of the MOCO problems considered: no equivalent solution generated will be thus retained. We are just interesting to approximate the Pareto front, that is for each non-dominated point, to find one solution that allows to generate this point. To find a minimal complete set is equivalent for a single-objective problem to generate only one optimal solution and not all optimal solutions.

2.1.3 ϵ-dominance

An interesting extension of the dominance relation of Pareto is the ϵ-dominance [255], where an ϵ factor is introduced in the dominance relation of Pareto.

Let y^1 and $y^2 \in \mathbb{R}^p$.

Definition 15 *ϵ-dominance: we say that y^1 ϵ-dominates y^2 if, and only if, $y^1_k \leq \epsilon \cdot y^2_k \; \forall k \in \{1, \ldots, p\}$, with $\epsilon \in \mathbb{R}^+$. We denote this relation by $y^1 \preceq_\epsilon y^2$.*

Roughly speaking, a vector y^1 is said to ϵ-dominates another vector y^2 if, when each component of y^2 is multiplied by a positive factor of ϵ, the resulting objective vector is still weakly dominated by y^1.

In the case of y^1 and $y^2 \in \mathbb{R}^p_{\succeq}$ it is interesting to look at the *minimal* value that ϵ can take to keep the ϵ-dominance. Indeed, it is always possible to use for ϵ a high value superior to 1 so that the ϵ-dominance occurs. We will call $\epsilon_{\min}$ the minimal value for ϵ.

To have the ϵ-dominance between y^1 and y^2, that is $y^1 \preceq_\epsilon y^2$, the following inequalities must be satisfied:

$$y^1_k \leq \epsilon \cdot y^2_k \Leftrightarrow \epsilon \geq \frac{y^1_k}{y^2_k} \ \forall k \in \{1, \ldots, p\}$$

The minimal value for ϵ is thus equal to $\max_{k=1,\ldots,p} \left(\frac{y^1_k}{y^2_k}\right)$.

We divide the different possibilities in four cases:

1. If $y^1 \prec\prec y^2$ then $0 < \epsilon_{\min} < 1$, since $y^1_k < y^2_k \ \ \forall k \in \{1, \ldots, p\}$.
2. If $y^1 \prec y^2$ but not $y^1 \prec\prec y^2$, then $\epsilon_{\min}$ is equal to 1, since for at least one $k \in \{1, \ldots, p\}$ we have $y^1_k = y^2_k$.
3. If $y^1 = y^2$ then $\epsilon_{\min}$ is obviously equal to 1.
4. If $y^1 || y^2$ or $y^2 \prec y^1$ or $y^2 \prec\prec y^1$, then $\epsilon_{\min} > 1$, since for at least one $k \in \{1, \ldots, p\}$, we have $y^1_k > y^2_k$.

Therefore, if it exists at least one value for ϵ in the open interval $(0, 1)$ such that $y^1 \preceq_\epsilon y^2$, then $y^1 \prec\prec y^2$.

2.1.4 Lexicographic optimality

We introduce in this section, another definition of optimality, the lexicographic optimality. This notion of optimality is used when there is a ranking among the objectives. For example, if $p = 2$, we first try to generate an optimal solution for the first objective and if there is more than one optimal solution, we optimize the second objective under the constraint than the first objective has still an optimal value. This means that even an extremely good value for the second objective cannot compensate a slightly non-optimal first objective.

We now present a formal definition of the lexicographic optimality.

Let $y^1, y^2 \in \mathbb{R}^p$, and if $y^1 \neq y^2$, let $k^* := \min\{k : y^1_k \neq y^2_k\}$.

1. Lexicographic order: $y^1 \preceq_{lex} y^2 \Leftrightarrow y^1_{k^*} < y^2_{k^*}$ or $y^1 = y^2$.
2. Strict lexicographic order: $y^1 \prec\prec_{lex} y^2 \Leftrightarrow y^1_{k^*} < y^2_{k^*}$.

The $\preceq_{lex}$ relation is a total preorder (reflexive, antisymmetric, transitive, connected), that means that all solutions can be compared, but ties are possible.

The lexicographic optimization problem can be written as follows:

$$\operatorname*{lexmin}_{x \in \mathcal{X}} \big(f_1(x), \ldots, f_p(x)\big) \tag{2.4}$$

Definition 16 *Lexicographic optimal solution: a feasible solution $x^* \in X$ is called a* lexicographic optimal *solution (or is lexicographically optimal) if there does not exist any other feasible solution $x \in \mathcal{X}$ such that $f(x) \prec\prec_{lex} f(x^*)$.*

In addition to this definition, we can state that $x^* \in \mathcal{X}$ is a lexicographic optimal solution if

$$f(x^*) \preceq_{lex} f(x) \ \ \forall x \in \mathcal{X}.$$

It is interesting to note that a lexicographic optimal solution is an efficient solution:

Theorem 1 *Let $x^* \in \mathcal{X}$ be such that $f(x^*) \preceq_{lex} f(x) \ \forall x \in X$. Then $x^* \in X_E$.*

Algorithm 1 `Lexicographic optimal solutions`

Parameters ↓: Feasible set $\mathcal{X}$ and objective functions $f_1, f_2, \ldots, f_p$ of a MOP.
Parameters ↑: Set of lexicographic optimal solutions.
--| Initialization
$X_1 := \mathcal{X}$
$k := 1$
while $k \leq p$ **do**
  Solve the single-objective problems $\min_{x \in X_k} f_k(x)$ (1).
  if (1) has a unique optimal solution $\widehat{x}_k$ **then**
    STOP, $\widehat{x}_k$ is the unique optimal solution of the lexicographic optimization problem.
  if $k = p$ **then**
    STOP, the set of optimal solutions of the lexicographic optimization problem is

$$\left\{x \in X_p : f_p(x) = \min_{x \in X_p} f_p(x)\right\}$$

  Let $X_{k+1} := \left\{x \in X_k : f_k(x) = \min_{x \in X_k} f_k(x)\right\}$ and let $k := k + 1$.

See [53] for a proof.

An algorithm to find all the lexicographic solutions is given in Algorithm 1.

It is also possible to define a lexicographic solution for another ranking of the objectives than that given by the indexes, that is $f(x) = (f_1(x), f_2(x), \ldots, f_p(x))$. We call a solution that is lexicographically optimal for at least one ranking of the objectives a *global* lexicographic solution, defined as follows:

Definition 17 *Global lexicographic optimal solution: a feasible solution $x^* \in X$ is called a* global lexicographic *solution if there does not exist any other feasible solution $x \in X$ such that $f^\pi(x) \preceq_{lex} f^\pi(x^*)$ for some permutation f^π of $(f_1, \ldots, f_p)$.*

We show in Figure 2.6 the representation in objective space of the global lexicographic solutions of the example of Figure 2.2.

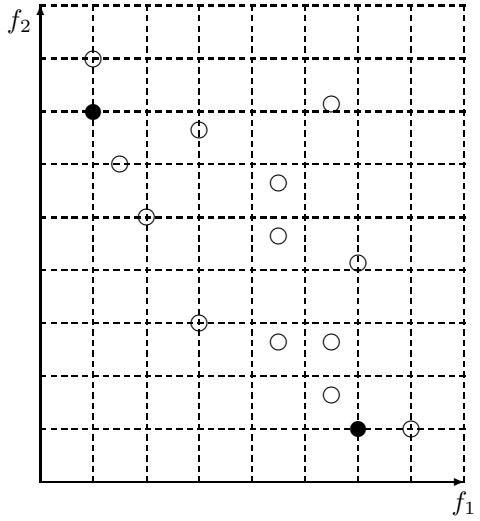

Figure 2.6: Representation of the global lexicographic solutions (one for $f = (f_1, f_2)$ and the other for $f = (f_2, f_1)$).

2.1.5 Ideal and nadir points

The ideal and nadir points are important concepts in MOO since these points allow to define lower and upper bounds of the Pareto front. These points give also the range of the values that the non-dominated points can take for each objective. The aim is to find two bounds $\underline{y}$ and $\overline{y}$, points in the objective space, such that $\underline{y}_k \leq y_k \leq \overline{y}_k \ \forall\, k \in \{1, \ldots, p\}$ and $\forall\, y \in Y_N$.

Good values for $\underline{y}_k$ can be generated as follows:

Definition 18 *Ideal point: we call y^I the ideal point, which has for coordinates in objective space, the best values for each objective:*

$$\underline{y}_k = y^I_k = \min_{x \in \mathcal{X}} f_k(x) = \min_{y \in Y_N} y_k \quad \forall\, k \in \{1, \ldots, p\} \tag{2.5}$$

This bound is tight since there is always a non-dominated point $y \in Y_N$ with $y_k = y^I_k$.

A possibility for $\overline{y}_k$ is as follows:

$$\overline{y}_k = \max_{x \in \mathcal{X}} f_k(x) = \max_{y \in Y} y_k \quad \forall\, k \in \{1, \ldots, p\} \tag{2.6}$$

But this lower bound tends to be far away from non-dominated points [53].

Therefore, a tighter lower bound, called the nadir point, can be generated as follows:

Definition 19 *Nadir point: we call y^N the nadir point which has for coordinates in the objective space, the worst values for each objective, for the solutions belonging to the efficient set:*

$$y^N_k = \max_{x \in X_E} f_k(x) = \max_{y \in Y_N} y_k \quad \forall\, k \in \{1, \ldots, p\} \tag{2.7}$$

The nadir point is thus defined as the maximum over non-dominated points only.

We show in Figure 2.7 the representation of the ideal point, nadir point and the basic upper bound $\overline{y}$ of the example of Figure 2.2.

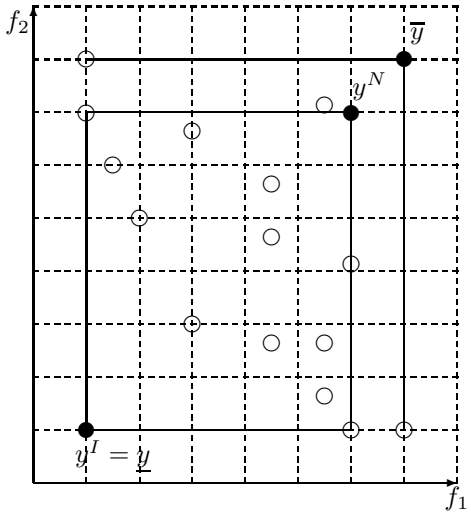

Figure 2.7: Representation of the ideal point, nadir point and the basic upper bound $\overline{y}$ ($p = 2$).

To generate the ideal point y^I, it is enough to solve p single-objective optimization problems. Its computation can be thus considered as easy (from a multicriteria point of view). On the other hand, the computation of the nadir point y^N involves optimization over the efficient set, which is not an easy task at all. As far as we know, there does not exist an efficient method to determine y^N for a general MOP. Heuristics are often used to produce a good approximation of the nadir point. A basic estimation of the nadir point can be realized by computing the *pay-off table* of the MOP. This technique works as follows [53].

First, p single-objective problems $\min_{x\in\mathcal{X}} f_k(x)$ are solved. Let the optimal solutions be x^k, $\forall\, k \in \{k = 1,\ldots,p\}$, that is $f_k(x^k) = \min_{x\in\mathcal{X}} f_k(x)$. We then compute the pay-off table as follows, by making use of these optimal solutions:

	x^1	x^2	$\cdots$	x^{p-1}	x^p
f_1	y_1^I	$f_1(x^2)$	$\ldots$	$f_1(x^{p-1})$	$f_1(x^p)$
f_2	$f_2(x^1)$	y_2^I	$\ldots$	$f_2(x^{p-1})$	$f_2(x^p)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
f_{p-1}	$f_{p-1}(x^1)$	$\ldots$	$\ldots$	$\ddots$	$f_{p-1}(x^p)$
f_p	$f_p(x^1)$	$f_p(x^2)$	$\ldots$	$f_p(x^{p-1})$	y_p^I

We can immediately get the ideal point from the pay-off table, since $y_k^I = f_k(x^k)\ \forall\, k \in \{1\ldots,p\}$.

The approximation of the nadir point is obtained as follows:

$$\tilde{y}_i^N := \max_{k=1\ldots p} f_i(x^k) \quad \forall\, i \in \{1\ldots,p\} \tag{2.8}$$

that is, the largest element in row i is considered as an estimate for y_i^N.

With this technique, as the solutions x^k are not necessary efficient (but are weakly efficient), overestimation of the nadir point is possible. On the other hand, if the solutions x^k are guaranteed to be efficient, overestimation is impossible but underestimation can occur (if $p > 2$). For an example, we refer the reader to [53].

In the case of $p = 2$, the value of y_2^N is attained when the first objective is optimal and vice versa, that is the value of of y_1^N is attained when the second objective is optimal. Therefore, the nadir point can be easily obtained by generating the two global lexicographic optimal solutions.

It should be noted that we have $y^I \neq y^N$, as we consider conflicting objective.

2.2 MOCO problems

We now present the specific concepts related to MOCO problems. Some of these concepts are also applicable to MOILP, but we focus the presentation on MOCO problems.

2.2.1 Supported - non-supported efficient solutions

In MOCO, there exists an important classification of the efficient solutions: *supported* efficient solutions and *non-supported* efficient solutions.

The images of the supported efficient solutions in objective space are located on the convex hull of the Pareto front and the images of the non-supported efficient solutions are not located on the convex hull of the Pareto front. More precisely, we can characterize these solutions as follows [53]:

- Supported efficient solutions: supported efficient solutions are optimal solutions of the following weighted sum single-objective problem

$$\min_{x\in\mathcal{X}} \sum_{k=1}^{p} \lambda_k\, f_k(x) \tag{2.9}$$

  for some vector $\lambda \succ\succ 0$, that is each component is positive ($\lambda_k > 0, \forall\, k \in \{1,\ldots,p\}$). The image in objective space of the supported efficient solutions, called supported non-dominated points, are located on the "lowerleft boundary" of the convex hull of $\mathcal{Y}$ (conv $(\mathcal{Y})$), that is they are non-dominated points of $\text{conv}(\mathcal{Y}) + \mathbb{R}^p_{\leq}$ (the nonnegative orthant of $\mathbb{R}^p$). We can obtain all supported efficient solutions by varying the weight set λ and by solving the corresponding weighted sum single-objective problems. Supported efficient solutions are denoted by X_{SE} and supported non-dominated points by Y_{SN}.

- Non-supported efficient solutions: non-supported efficient solutions are efficient solutions that are not optimal solutions of any weighted sum single-objective problem with $\lambda \succ\succ 0$. Non-supported non-dominated points are located in the interior of $\text{conv}(\mathcal{Y})+\mathbb{R}^p_{\preceq}$. Non-supported efficient solutions are denoted by X_{NE} and non-supported non-dominated points by Y_{NN}. In the case of $p = 2$, the non-supported non-dominated points are always located in the interior of the right triangle built between two consecutive supported non-dominated points.

We have represented in Figure 2.8 the supported non-dominated points (on the left) and the non-supported non-dominated points (on the right) of the example of Figure 2.2.

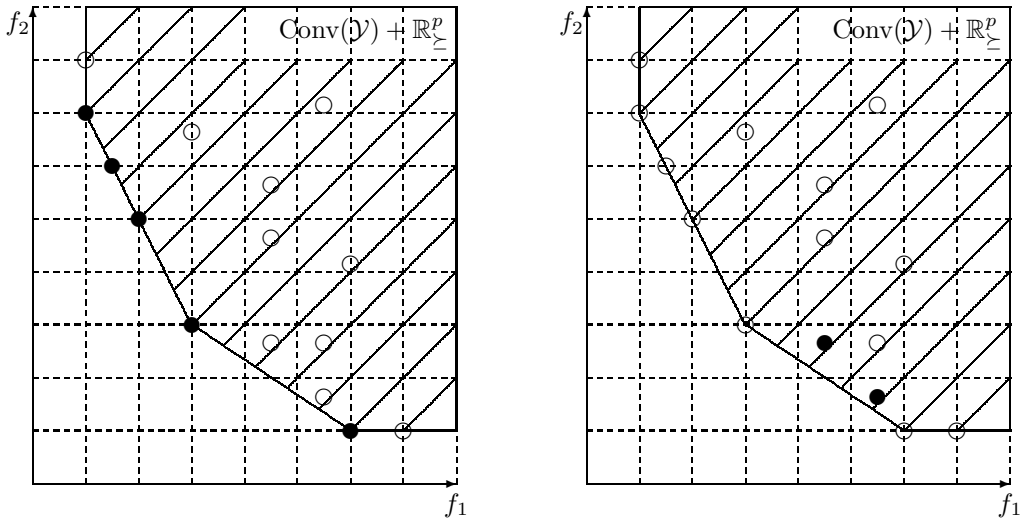

Figure 2.8: On the left: supported non-dominated points (Y_{SN}). On the right, non-supported non-dominated points (Y_{NN}). The first four supported non-dominated points have coordinates equal to respectively $(1, 7), (1.5, 6), (2, 5), (3, 3)$ and the solutions that correspond to these points are all optimal solutions of $\min\big(2f_1(x) + f_2(x)\big)$ (the optimal value is equal to 9). The last supported non-dominated point has for coordinates $(6, 1)$ and corresponds, like the point (3,3), to an optimal solution of $\min\big(2f_1(x) + 3f_2(x)\big)$ (the optimal value is equal to 15). On the other hand, the solutions that correspond to the two non-supported non-dominated points are not optimal solutions of any weighted sum problems. The supported efficient solutions are always better than the non-supported efficient solutions on weighted sum problems.

We can also make a distinction between supported efficient solutions and define extreme supported efficient solutions and non-extreme supported efficient solutions [53]:

- Extreme supported efficient solutions: the objective vectors $f(x)$ of these supported efficient solutions (called extreme supported non-dominated points) are located on the vertex set of $\text{conv}(\mathcal{Y})$, that is they are extreme points of $\text{conv}(\mathcal{Y}) + \mathbb{R}^p_{\preceq}$. We call the extreme supported efficient solutions X_{SE1} and the image $f(X_{SE1})$ of X_{SE1} in objective space is called Y_{SN1}.

- Non-extreme supported efficient solutions: the objective vectors $f(x)$ of these supported efficient solutions (called non-extreme supported non-dominated points) are not located on the vertex set of $\text{conv}(\mathcal{Y})$ but located in the relative interior of the faces of $\text{conv}(\mathcal{Y}) + \mathbb{R}^p_{\preceq}$. This set is denoted by X_{SE2} and the image $f(X_{SE2})$ of X_{SE2} in objective space by Y_{SN2}.

We have represented in Figure 2.9 the extreme supported non-dominated points (on the left) and the non-extreme supported non-dominated points (on the right) of the example of Figure 2.2.

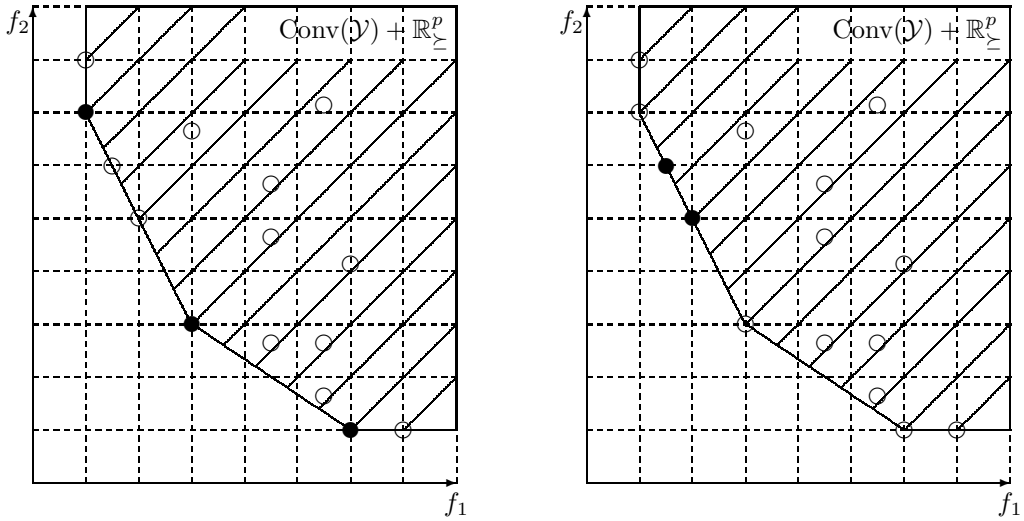

Figure 2.9: On the left: extreme supported non-dominated points Y_{SN1}. On the right, non-extreme supported non-dominated points Y_{SN2}.

We can wonder whether the non-supported efficient solutions are worthwhile in practice since these solutions are always worse than supported efficient solutions on weighted sum problems.

In other words, is it useful to produce these solutions?

Let us consider the following elementary MOCO problem [228], corresponding to a biobjective knapsack problem (see section 2.2.4):

$$\begin{aligned} \max f_1(x) = \quad & 6x_1 + 3x_2 + x_3 \\ \max f_2(x) = \quad & x_1 + 3x_2 + 6x_3 \\ \text{subject to} \quad & x_1 + x_2 + x_3 \leq 1 \\ & x_i \in \{0,1\} \qquad i = 1, \ldots, 3 \end{aligned} \tag{2.10}$$

For this problem $\mathcal{X} = X_E = \{(1,0,0),(0,1,0),(0,0,1)\}$ and $\mathcal{Y} = Y_N = \{(6,1),(3,3),(1,6)\}$. If we represent the non-dominated points (see Figure 2.10), we see that the point (3,3) is non-supported while the points (1,6) and (6,1) are supported. Therefore, the only non-dominated point establishing a good compromise between both objectives is non-supported.

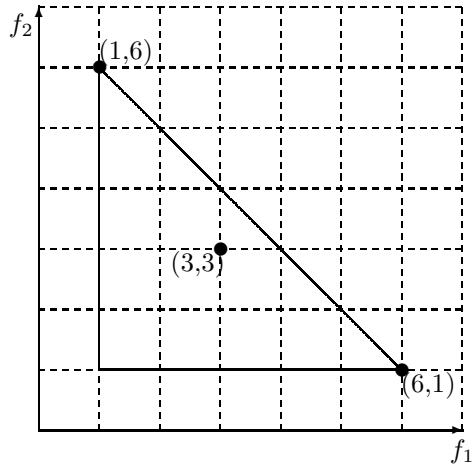

Figure 2.10: Representation of the non-dominated points of the problem (2.10). The only point establishing a good compromise between both objectives is the non-supported point (3,3).

We can also wonder whether the number of non-supported efficient solutions is high in comparison with the number of supported efficient solutions. There is no theoretical answer to this question, but, for many MOCO problems, we often observe that the number of supported efficient solutions grows *polynomially* with instance size while the number of non-supported efficient solutions grows *exponentially* with instance size. We will observe that in the chapters 7 and 8 concerning the experimentations.

We have represented in Figure 2.11 an example summarizing the different types of efficient solutions presented, in the decision and objective space, as well as an example of a minimal complete set.

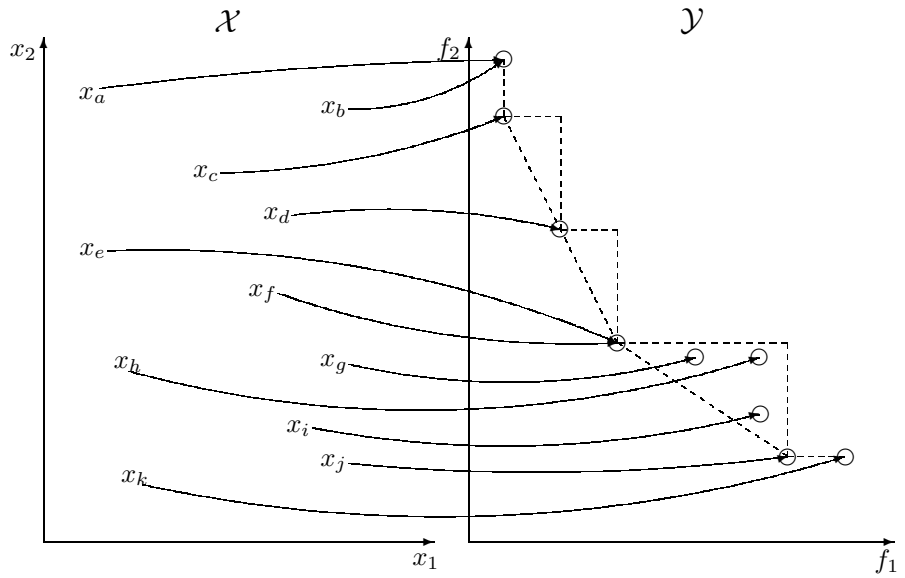

Figure 2.11: Representation in decision space (with an arbitrary dimension of two) and objective space of different types of efficient solutions. From this figure: $X_{wE} = \{x_a, x_b, x_c, x_d, x_e, x_f, x_g, x_h, x_i, x_j, x_k\}$; $X_E = \{x_c, x_d, x_e, x_f, x_g, x_i, x_j\}$; $X_{SE} = \{x_c, x_d, x_e, x_f, x_j\}$; $X_{SE1} = \{x_c, x_e, x_f, x_j\}$; $X_{SE2} = \{x_d\}$; $X_{NE} = \{x_g, x_i\}$; a minimal complete set $= \{x_c, x_d, x_e, x_g, x_i, x_j\}$.

2.2.2 Complexity and intractability of MOCO problems

In this section, we will talk about the complexity and the intractability of MOCO problems.

The difficulty to generate the efficient set comes essentially from these two issues:

- Difficult to check if a solution is efficient (complexity).
- The number of efficient solutions can be very large (intractability).

The computational complexity theory is a large field that investigates the difficulty in providing algorithms that are efficient algorithms for both general and specific computational problems. It is the theory of "how difficult" is it to answer a decision problem DP, where a decision problem is a question that has either a "yes" or "no" answer. This difficulty is measured by the number of operations an algorithm needs to find the correct answer to the decision problem in the worst case (worst case analysis).

The decision version of a single-objective optimization problem is the following question: given a constant $b \in \mathbb{Z}$, does there exists $x \in X$ such that $f(x) \leq b$?

Roughly speaking, and without entering into the details (we refer to [80, 219] for more details) a decision problem belongs to the class $\mathcal{P}$ of problems, if there exists a deterministic algorithm

that answers to the decision problem and needs $\mathcal{O}\big(p(n)\big)$ operations, where p is a polynomial in n (number of operations performed by the algorithm is less than or equal to $c \cdot p(n)$ where c is a constant). On the other hand, a decision problem belongs to the class $\mathcal{NP}$ if there is a nondeterministic polynomial time algorithm that solves the decision problem[1]. We have thus $\mathcal{P} \subset \mathcal{NP}$. A decision problem DP belongs to the class $\mathcal{NP}$-Complete if DP $\in \mathcal{NP}$ and for all DP' in $\mathcal{NP}$, there exists a polynomial time transformation from DP' to DP. In other words, all DP' are "special cases" of DP, and therefore DP is at least as difficult as DP'. Optimization problems whose decision problems are $\mathcal{NP}$-Complete are called $\mathcal{NP}$-Hard.

For an instance I of a minimization MOP, the multiobjective decision problem associated consists, given an integer vector $b = (b_1, \ldots, b_p)$, to decide if it exists a feasible solution of I such that the value on each criterion i is less or equal to b_i ($\forall\, k \in \{k = 1, \ldots, p\}$).

We present in Table 2.1 some results about the complexity of some classic CO problems in their single and MO versions [208].

Combinatorial Problem	$p = 1$	$p \geq 2$
Shortest path	Polynomial	$\mathcal{NP}$-Hard
Minimum spanning tree	Polynomial	$\mathcal{NP}$-Hard
Assignment	Polynomial	$\mathcal{NP}$-Hard
Minimal cut s-t	Polynomial	$\mathcal{NP}$-Hard
Knapsack	$\mathcal{NP}$-Hard	$\mathcal{NP}$-Hard
Traveling salesman	$\mathcal{NP}$-Hard	$\mathcal{NP}$-Hard

Table 2.1: Results of complexity of some classic CO problems, single and MO versions.

We remark that some CO problems for which there exists a polynomial algorithm in their single-objective version are $\mathcal{NP}$-Hard in their multiobjective version. This is the case for the shortest path, minimum spanning tree, assignment and minimal cut s-t problems. The knapsack problem and traveling salesman problems that are $\mathcal{NP}$-Hard in their single-objective version remains obviously $\mathcal{NP}$-Hard in their MO counterparts.

Let us take a look now at the number of efficient solutions of a MOCO problem. We first define the notion of intractability [53].

Definition 20 *Intractability: A MOCO problem is called intractable if the number of non-dominated points can be exponential in the size of the instance.*

A simple example of intractable MOCO problem is given by Ehrgott in [53]. This MOCO problem is called the unconstrained problem and is defined as follows:

$$\begin{aligned} \min \sum_{i=1}^{n} c_i^k x_i \quad & k = 1 \ldots p \\ \text{subject to } x_i \in \{0,1\} \quad & i = 1 \ldots n \end{aligned} \tag{2.11}$$

We can construct an intractable instance by setting $c_i^k = (-1)^k 2^{i-1}$. In this case, we obtain :

$$\mathcal{Y} = Y_N = \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -2 \\ 2 \end{pmatrix}, \begin{pmatrix} -3 \\ 3 \end{pmatrix}, \ldots, \begin{pmatrix} -2^n + 1 \\ 2^n - 1 \end{pmatrix} \right\}$$

and $\mathcal{X} = X_E$.

Therefore, $|Y_N| = 2^n$, which means that the number of non-dominated points is exponential in the size of an instance. The unconstrained problem is thus intractable.

[1] This is done in two stages. The first stage ("guessing" stage) consists in generating as many solutions as we want while the second stage ("checking" stage) consists in checking, for all generated solutions x, that $x \in \mathcal{X}$ and $f(x) \leq b$ in polynomial time.

It does not exist a proof showing that every MOCO problem is intractable but most of the classic MOCO problems have been proved to be intractable (shortest path, minimum spanning tree, assignment, minimal cut s-t, traveling salesman, ... are intractable).

To summarize, MOCO problems are in the worst case $\mathcal{NP}$-Hard, with a non-convex feasible set $\mathcal{X}$ and a huge number of efficient solutions!

2.2.3 Connectedness of efficient solutions

The connectedness of efficient solutions is of particular interest, since if this is true, once an efficient solution has been found, a local search starting from this solution allows to find all other efficient solutions. In other words, it would exist a path in the decision space between all efficient solutions, and with the neighborhood of the local search we would be able to travel through this path and to generate all efficient solutions.

The notion of connectedness of efficient solutions has been introduced by Ehrgott and Klamroth [61] in 1997 and under the related notion of *global convexity* by Borges and Hansen [23] in 2000. By global convexity, it is understood that in MOCO problems, the efficient solutions are concentrated in a small fraction of decision space and those are neighbors in objective space are also neighbors in decision space.

The paper of Gorski *et al.* [93] summarizes all known results concerning the connectedness of efficient solutions and new results for different MOCO problems are exposed. The rest of this section is mainly based on this paper.

Due to the discrete structure of X_E, a graph-theoretical model is used to define the connectedness of efficient solutions [61, 188]:

Definition 21 *For a given MOCO problem the* adjacency graph *of efficient solutions* $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ *of the MOCO problem is defined as follows:* $\mathcal{V}$ *consists of all efficient solutions of the given MOCO problem. An (undirected) edge is introduced between all pairs of vertices which are adjacent with respect to the considered definition of adjacency for the given problem. These edges form the set* $\mathcal{A}$*.*

We can now define the connectedness of X_E via the adjacency graph.

Definition 22 *The set* X_E *of all efficient solutions of a given MOCO problem is said to be connected if its corresponding adjacency graph* $\mathcal{G}$ *is connected.*

We still have to define the notion of adjacency for a given problem.

For combinatorial problems, the notion of adjacency is problem-dependent and usually based on *elementary moves* which transform one feasible solution into another feasible solution; these two solutions are then adjacent. An elementary move is called *efficient* if it leads from one efficient solution of the problem to another efficient solution.

Examples of elementary moves are the exchange of two edges for the traveling salesman problem, the insertion and deletion of edges in a spanning tree, or simply the swap of two bits in a binary solution vector.

A *canonical* move is an elementary move for a given problem class if the set of optimal solutions of the corresponding single-objective problem is connected for all problem instances. If for a MOCO problem, the elementary move is not canonical in the single-objective counterpart of the problem, it implies non-connectedness.

It is also possible to use a more universal and less problem dependent adjacency concept which utilizes MILP formulations of MOCO problems and based on the topologically adjacency defined by Isermann [108]. We refer the reader to [92] for more details about this concept.

Ehrgott and Klamroth [61] showed that the adjacency graphs of efficient shortest paths and of efficient spanning trees are non-connected in general. Gorski *et al.* [93] extended these results and showed that the adjacency graphs of weakly efficient shortest paths and weakly efficient spanning trees are non-connected in general.

Gorski *et al.* [93] have also presented new results and showed that the adjacency graph of a binary multiple choice knapsack problem with equal weights, of a binary knapsack where the

number of items in each solution is fixed to a constant, of a multiple objective unconstrained binary problem, of the biobjective linear assignment problem and of the traveling salesman problem are non-connected in general.

Therefore, the adjacency graph of many MOCO problems have been shown as non-connected. However, this result does not call the use of local search in question since the aim of a local search is not to find all efficient solutions. Moreover, in practice, not all but many efficient solutions are connected, and it is always possible to start the local search several times from different solutions located in the connected part of the adjacency graph.

We present now the two classical MOCO problems that will be studied in this work.

2.2.4 Multiobjective multidimensional knapsack problem

The knapsack problem (KP) consists, in its unidimensional and unicriteria case, in choosing a set of objects to put in a bag. Each object having a weight and a profit, the objects should be selected to maximize the total profit while not exceeding the bag capacity. The single-objective version of this problem has been studied extensively in the literature [133, 168].

In the MO case, p profits are associated to each object and in the multidimensional case, m constraints are considered. The multidimensional multiobjective knapsack (MOMKP) problem is defined as follows:

$$\begin{aligned} \text{“max”} \quad & f_k(x) = \sum_{i=1}^{n} c_k^i x_i && k = 1, \ldots, p \\ \text{subject to} \quad & \sum_{i=1}^{n} w_j^i x_i \leq W_j && j = 1, \ldots, m \\ & x_i \in \{0,1\} && i = 1, \ldots, n \end{aligned} \tag{2.12}$$

with n the number of objects available, x the decision vector, formed of the binary variables x_i ($x_i = 1$ means that the object i is in the knapsack), c_k^i the profit k of the object i, w_j^i the weight j of the object i and W_j the capacity j of the knapsack.

It is assumed that all coefficients c_k^i, w_j^i and W_j are nonnegative.

The KP problem is known to be $\mathcal{NP}$-Hard [181]. As a consequence, the MOMKP is also $\mathcal{NP}$-Hard. For the intractability, no result is known [53]. Indeed, counting the number of feasible solutions of a MKP is not easy [52].

A simple, important and famous result for the single-objective unidimensional KP is presented below. In 1957, Dantzig [40] showed that it is easy to obtain an optimal solution of the linear relaxation of the KP. It is indeed enough to sort the items according to their non-increasing profit-to-weight ratios (also called efficiencies), and adding the items to the knapsack until the knapsack is full, and to insert a fraction of the last item that cannot entirely be in the knapsack. This item b, is called the break or critical item. We can then define an optimal solution in the following way:

If the items are ordered such that:

$$\frac{c^1}{w^1} \geq \ldots \geq \frac{c^j}{w^j} \geq \ldots \frac{c^n}{w^n}$$

an optimal solution $\widehat{x}$ of the linear relaxation of the single-objective KP is obtained as follows:

$$\widehat{x}_i = \begin{cases} 1 & \text{if } i < b \\ \frac{W - \sum_{j=1}^{b-1} w^j}{w^b} & \text{if } i = b \\ 0 & \text{if } i > b \end{cases}$$

where the item b is defined by:

$$\sum_{i=1}^{b-1} w^i \leq W < \sum_{i=1}^{b} w^i$$

In 1980, Balas and Zemel [13] noticed that the optimal solution for random instances of the KP was very close to the optimal solution of the linear relaxation of the KP. The notion of *core* has thus been defined. Assuming that x^* is an optimal solution of the KP and that the items are sorted according to their non-increasing efficiencies, the core is a set of items C equal to $\{i_1, \ldots, i_2\}$, where $i_1 = \min\{i : x_i^* = 0,\ i = 1, \ldots, n\}$ and $i_2 = \max\{i : x_i^* = 1,\ i = 1, \ldots, n\}$. As the size of the core has been shown as small in regard to the total number of items, the core concept is the underlying concept in the development of the most efficient known algorithms for solving the KP.

2.2.5 Multiobjective traveling salesman problem

The traveling salesman problem (TSP) is certainly the best-known and most studied $\mathcal{NP}$-Hard single-objective CO problem. The book of Gutin and Punnen [96] analyzes various methods for the TSP and for some of its variants; the chapters of Johnson and McGeoch [124] and Johnson *et al.* [122] are devoted to heuristics respectively for the symmetric and asymmetric versions of the TSP. Other interesting reviews are related to the use of metaheuristics for the TSP: [123] is a survey of local search (LS) methods, [197] of local search and metaheuristics, [145] of genetic algorithms (GA) and [171] of memetic algorithms applied to the TSP.

We define the MOTSP as follows.

Let $\mathcal{K}_n = (\mathcal{V}, \mathcal{E})$ be a complete graph with the set of nodes $\mathcal{V} = \{1, \ldots, n\}$ and the set of edges $\mathcal{E}$. The nodes stand for cities, and the edges for paths connecting the cities. At each edge $e \in \mathcal{E}$, a vector of cost $c(e)$ is associated, defining p criteria to minimize. The criteria considered can correspond to cost, duration, distance, safety, the beauty of the scenery, etc.

The salesman starts a tour from one city, and has to visit all the cities exactly once and then to return to the city from which he started his tour. Such a tour corresponds to a *Hamiltonian* cycle (see Figure 2.12).

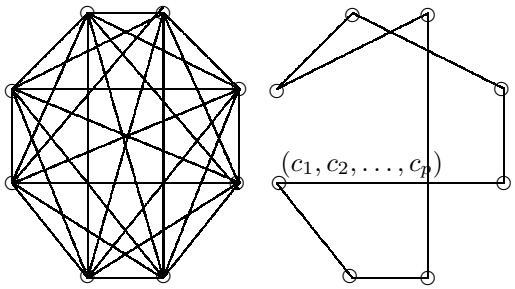

Figure 2.12: Example of an instance of the MOTSP (represented by a complete graph) and of a Hamiltonian cycle (n=8, p objectives).

The optimization problem is to find a Hamiltonian cycle of minimal cost (by considering the p objectives).

If we called by δ a Hamiltonian cycle on $\mathcal{K}_n$ and by $\mathcal{F}$ the set of all Hamiltonian cycles, we can formulate the MOTSP as follows:

$$\text{“min”}\, f_k(\delta) = \sum_{e \in \delta} c_k(e) \text{ with } \delta \in \mathcal{F} \quad k = 1, \ldots, p \tag{2.13}$$

More precisely, as a Hamiltonian cycle gives the order in which visiting the nodes, a tour can be represented by $\delta(1), \delta(2), \ldots, \delta(n), \delta(1)$. Let $e = (i, j)$ be the edge between the nodes i and

j and $c_k(i,j)$ $\big(= c_k(e)\big)$ the value of the objective k between the nodes i and j. The objective functions of the MOTSP can be thus stated as:

$$\text{“min”} f_k(\delta) = \sum_{i=1}^{n-1} c_k(\delta(i), \delta(i+1)) + c_k(\delta(n), \delta(1)) \text{ with } \delta \in \mathcal{F} \quad k = 1, \ldots, p \tag{2.14}$$

The mathematical formulation of the MOTSP often considered uses the binary decision variable $x_{i,j}$, defined as follows:

$$x_{i,j} = \begin{cases} 1 & \text{if the edge } (i,j) \text{ is used,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.15}$$

The mathematical formulation of the MOTSP is then the following:

$$\text{“min”} f_k(\delta) = \sum_{(i,j)\in\mathcal{E}} c_k(i,j)\, x_{i,j} \quad k = 1, \ldots, p \tag{2.16}$$

$$\sum_{i=1}^{n} x_{i,j} = 1 \quad j = 1, \ldots, n \tag{2.17}$$

$$\sum_{j=1}^{n} x_{i,j} = 1 \quad i = 1, \ldots, n \tag{2.18}$$

$$\sum_{(i,j)\in\mathcal{E}, i\in V, j\in\mathcal{V}\setminus V} x_{i,j} \geq 1 \text{ for } 2 \leq |V| \leq n-2 \text{ with } V \subseteq \mathcal{V} \tag{2.19}$$

$$x_{i,j} \in \{0,1\} \text{ with } (i,j) \in \mathcal{E} \tag{2.20}$$

The constraint (2.17) ensures that the tour *enters* node in j exactly once, for each node $j \in \mathcal{V}$.

The constraint (2.18) ensures that the tour *leaves* node i exactly once, for each node $i \in \mathcal{V}$.

The constraint (2.19) allows to eliminate all possible subtours. These constraints are often called *subtour elimination constraints*. There are other alternative formulations to define the subtour elimination constraints, see [96] for a discussion about that.

The last constraint (2.20) defines the decision variables as binary.

The multiobjective TSP is $\mathcal{NP}$-Hard since the single-objective counterpart is.

Concerning the intractability, we have the following result [64]:

Theorem 2 *The multiobjective TSP is intractable, even if* $p = 2$.

See [53] for a proof.

We show now that the adjacency graph of the MOTSP is not connected.

As not appropriate MILP formulation of the problem is available, a MILP-based definition adjacency is not immediate. Paquete *et al.* [184] and Paquete and Stützle [188] have thus introduced a combinatorial definition of adjacency for the MOTSP: two feasible tours are adjacent if they differ in exactly four edges. This definition corresponds to the classic two-edge exchange neighborhood. However, this elementary move is not canonical.

Proof 1 [93] Considering the following instance of a symmetric TSP with five nodes and cost matrix C given by:

$$D = \begin{pmatrix} 0 & 7 & 10 & 3 & 1001 \\ 7 & 0 & 3 & 10 & 1000 \\ 10 & 3 & 0 & 7 & 1000 \\ 3 & 10 & 7 & 0 & 1001 \\ 1001 & 1000 & 1000 & 1001 & 0 \end{pmatrix}$$

The problem has two optimal tours $\delta_1 = (1,4,3,2,5,1)$ and $\delta_2 = (1,4,5,3,2,1)$ with evaluations $f(\delta_1) = f(\delta_2) = (3 + 7 + 3 + 1000 + 1001) = (3 + 1001 + 1000 + 3 + 7) = 2014$. But δ_1 differs from δ_2 in six edges, that is, δ_1 is not in the two-edge exchange neighborhood of δ_2.

3

Overview of methods for solving MOCO problems

3.1 Introduction

Many surveys of methods for solving MOCO problems have already been published. The literature on this subject is very plentiful, especially concerning metaheuristics, as testifies the website `http://www.lania.mx/~ccoello/EMOO/EMOObib.html` managed by C. Coello Coello that includes a vast list (more than 4000 of entries in August 2009) of bibliographic references on this field (originally dedicated to evolutionary multiobjective optimization, but that contains references on broader subjects).

With this chapter, we do not claim to replace these surveys but to simply introduce the basis for the understanding of the next chapters, especially the chapters 4 and 5 where surveys about the methods applied to respectively the MOMKP and MOTSP are presented (that are, as far as we know, the first works that review the solving methods of these specific problems).

We first present, in section 3.2, some general exact methods. For solving $\mathcal{NP}$-Hard problems, many heuristic methods have been developed. There are presented in the two next sections. In section 3.3, we say some words about approximation algorithms, that is heuristic methods with guarantee of performance. We then present the main section (section 3.4) of this chapter, the methods based on metaheuristics. As these last methods do not guarantee any performance, we will need tools and indicators in order to assess the quality of the approximations generated. That will be presented in section 3.5.

3.2 Exact methods

We first give theoretical results concerning the scalarization methods. We then introduce two typical exact methods for solving MOCO problems: the ϵ-constraint method and the two-phase method.

3.2.1 Scalarization methods

Weighted linear scalarizing functions

The weighted linear scalarizing function technique (or informally called the weighted sum) is probably the simplest and the most natural way to solve MOPs. However, for MOCO problems, for which the set of non-dominated points is generally non-convex, this method can only give an approximation of the efficient set X_E, composed of the supported efficient solutions.

In the weighted sum method, the MOP is transformed into a single-objective optimization problem by aggregating the objectives with a weighted sum:

$$\min_{x \in \mathcal{X}} \sum_{k=1}^{p} \lambda_k f_k(x) \quad (\text{with } \lambda \in \mathbb{R}^p_{\succ}) \tag{3.1}$$

We have the following theorems summarizing the results obtained by Geoffrion [82] for the problem (3.1):

Theorem 3 *Let $\widehat{x}$ be an optimal solution of (3.1):*

- *If $\lambda \succ\succ 0$ then $\widehat{x} \in X_{SE}$.*
- *If $\lambda \succ 0$ then $\widehat{x} \in X_{wSE}$.*
- *If $\lambda \succ 0$ and $\widehat{x}$ is a unique optimal solution then $\widehat{x} \in X_{SE}$.*

Theorem 4 *Let $\mathcal{X}$ and f such that $\mathcal{Y} = f(\mathcal{X})$ is convex (therefore in this case $X_E = X_{SE}$):*

- *If $\widehat{x} \in X_{wSE}$ then there is some $\lambda \succ 0$ such that $\widehat{x}$ is an optimal solution of (3.1).*
- *If $\widehat{x} \in X_{SE}$ then there is some $\lambda \succ\succ 0$ such that $\widehat{x}$ is an optimal solution of (3.1).*

Non-supported efficient solutions cannot be obtained with the weighted sum, since it is always possible to find a supported efficient solution better than a non-supported efficient solution for the problem (3.1).

Weighted Tchebycheff scalarizing function

One way to generate non-supported efficient solutions by considering scalarization functions is to use the weighted Tchebycheff scalarizing function (whose theoretical foundations have been assigned to [26]).

Let $||f(x) - y^0||_\lambda$ the λ-weighted Tchebycheff distance from $f(x) = (f_1(x), \ldots, f_p(x))$ to y^0, a reference point in objective space such that $y^0 \leq f(x)$, $\forall x \in \mathcal{X}$. Hence, $||f(x) - y^0||_\lambda = \max\limits_{k=1,\ldots,p} \left\{\lambda_k(f_k(x) - y^0_k)\right\}$.

The weighted Tchebycheff single-objective problem is defined as follows:

$$\min_{x \in \mathcal{X}} ||f(x) - y^0||_\lambda \tag{3.2}$$

We have the following theorem, coming from the chapter 14 of [214]:

Theorem 5 *Let $\widehat{x}$ be an optimal solution of (3.2):*

- *If $\lambda \succ 0$ then $\widehat{x} \in X_{wE}$.*
- *If $\widehat{x} \in X_E$ then there is some $\lambda \succ 0$ such that $\widehat{x}$ is an optimal solution of (3.2).*

We see that the convex property assumption is not anymore needed for the weighted Tchebycheff scalarizing function and with this technique it is thus possible to generate all the efficient solutions by correctly varying the weight set λ.

However, in spite of the interesting theoretical properties of the weighted Tchebycheff scalarizing function, this technique has been slightly used for solving MOCO problems. The main reasons are related to the difficulties to solve (3.2) and to the fact that the optimal solution of (3.2) is only weakly efficient.

Moreover, there is a certain lack of knowledge about the weighted Tchebycheff scalarizing function, as pointed out by Jaszkiewicz in his work about the adaptation of a metaheuristic (the MOGLS method, presented in section 3.4.3 of this chapter) to MOCO problems. In [114], Jaszkiewicz uses weighted Tchebycheff functions to adapt MOGLS to the MOMKP. On the other hand, in [118], he uses simple weighted sum functions to adapt MOGLS to the MOTSP. In this last work, he shows that for the MOTSP, it is more relevant to use weighted sum functions than weighted Tchebycheff functions for the following reasons:

- Weighted Tchebycheff functions are more difficult to optimize than weighted sum functions for the MOTSP.

- He ran experiments to show that weighted sum functions generate more potentially efficient solutions than weighted Tchebycheff functions for the same number of weight sets. He also showed that the solutions of the weighted sum functions are dispersed over the Pareto front while solutions of weighted Tchebycheff functions are more concentrated in the middle of the Pareto front.

His results confirm the following conclusion of Hansen [99] obtained earlier: in the case of the MOTSP, local search gives better final values for weighted Tchebycheff functions, when the search within the neighborhood of the current solution is guided by the weighted sum than when it is guided by the weighted Tchebycheff function.

The conclusion of Jaszkiewicz is that since the non-dominated set of the MOTSP is relatively smooth, weighted sum functions work better and weighted Tchebycheff functions should work better for problems with more irregular non-dominated sets.

It should be noted that Dächert and Gorski [41] have recently undertaken works on this subject. They study the influence of the reference point of weighted Tchebycheff functions on the results for MOCO problems. However, the conclusions that they obtained are not yet published.

Augmented weighted Tchebycheff scalarizing function

The augmented weighted Tchebycheff scalarizing function is an extension of the classic Tchebycheff function.

If we denote by $F(\lambda, x)$ the λ-weighted Tchebycheff distance to the ideal point y^I, that is $F(\lambda, x) = ||f(x) - y^I||_\lambda$, the augmented weighted Tchebycheff scalarizing function is defined as follows:

$$\min_{x \in \mathcal{X}} F(\lambda, x) + \rho \sum_{k=1}^{p} f_k(x) \tag{3.3}$$

with ρ small positive.

The advantage of the augmented Tchebycheff function is that, contrary to the classical Tchebycheff function, the optimal solution of (3.3) is efficient. However, the optimization of (3.3) is $\mathcal{NP}$-Hard [53].

3.2.2 ϵ-constraint

The ϵ-constraint is a well-known method to solve MOPs. The method has been introduced by Haimes *et al.* in 1971 [97]. In this method, there is no aggregation of the objectives: one objective is minimized while the remaining objectives are put as constraints.

The ϵ-constraint problem is defined as follows:

$$\begin{aligned} & \min_{x \in \mathcal{X}} f_j(x) \\ \text{subject to} \quad & f_k(x) \leq \epsilon_k \qquad k = 1, \ldots, p \quad k \neq j \end{aligned} \tag{3.4}$$

where $\epsilon \in \mathbb{R}^p$.

Theorem 6 *Let $\widehat{x}$ be an optimal solution of (3.4) for some j. Then $\widehat{x} \in X_{wE}$.*

(For a proof, see [53]).

We give (see Algorithm 2), in the case of $p = 2$ and $\mathcal{Y} \subset \mathbb{N}^2$, the algorithm based on the problem (3.4) that allows to generate all the efficient solutions (note that since the solution of (3.4) can be weakly efficient, lexicographic optimization is used). No convexity assumption is needed. As we work on $\mathbb{N}^2$, the value of the constraint on the second objective is fixed with a shift of one unit at each iteration. The $\geq$ constraint on the first objective is unnecessary but can help the lexicographic optimization.

The functioning of this algorithm is illustrated in Figure 3.1.

Algorithm 2 ϵ-constraint $(p = 2)$

$\epsilon_1 = -\infty$
$\epsilon_2 = +\infty$
repeat
  Solve $\underset{x \in \mathcal{X}}{\text{lexmin}}\big(f_1(x), f_2(x)\big)$ subject to $(f_1(x) \geq \epsilon_1, f_2(x) \leq \epsilon_2)$
  if $x \neq \varnothing$ **then**
    $\epsilon_1 = f_1(x) + 1$
    $\epsilon_2 = f_2(x) - 1$
until $x = \varnothing$

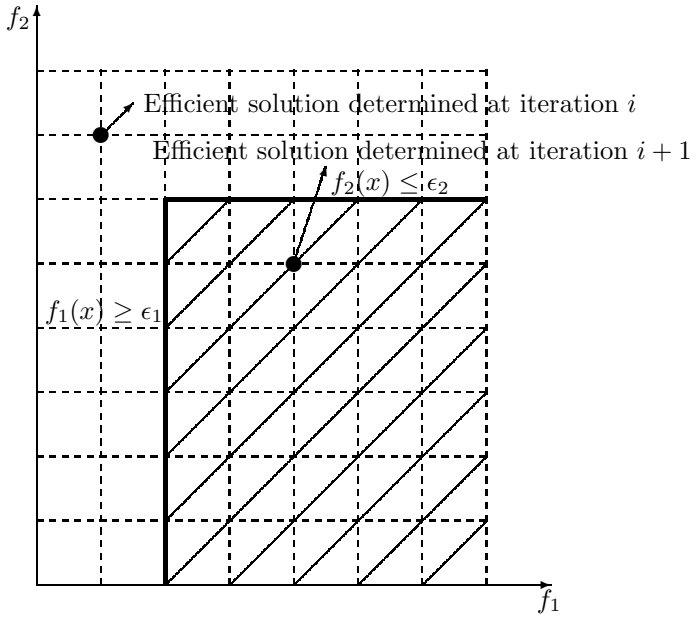

Figure 3.1: Illustration of the functioning of the Algorithm 2 (ϵ-constraint method) in the case of $p = 2$ and $Y \subset \mathbb{N}^2$.

As the addition of constraints modifies the structure of the problem, the efficiency of the method greatly depends on the quality of the procedure that solves (3.4). Exact commercial solvers are often used to solve this problem.

3.2.3 Two-phase method

The two-phase method has been developed by Ulungu and Teghem in 1995 [231]. In the original version, the method is only dedicated to solve biobjective problems. We present this version.

As a reminder, we use the following notations:

- $X_{E_{(m)}}$: (minimal) complete set of efficient solutions.
- $X_{SE_{(m)}}$: (minimal) complete set of supported efficient solutions.
- $X_{SE1_{(m)}}$: (minimal) complete set of extreme supported efficient solutions.
- $X_{SE2_{(m)}}$: (minimal) complete set of non-extreme supported efficient solutions.
- $X_{NE_{(m)}}$: (minimal) complete set of non-supported efficient solutions.

And $X_{E_{(m)}} = X_{SE_{(m)}} + X_{NE_{(m)}} = X_{SE1_{(m)}} + X_{SE2_{(m)}} + X_{NE_{(m)}}$.

The two phases are the following:

- Phase1: Generation of all supported efficient solutions (extreme and non-extreme ones) (X_{SE}).
- Phase2: Generation of all non-supported efficient solutions (X_{NE}).

Phase 1

The phase 1 is based on the Geoffrion's theorem [82] stating than an optimal solution of the problem (3.1) with $\lambda \succ\succ 0$ is a supported efficient (see theorem 3 of section 3.2.1).

The aim of phase 1 is to generate X_{SE} or eventually X_{SE_m} since X_{SE_m} is enough to generate all supported non-dominated points Y_{SN}.

The phase 1 is based on the dichotomic scheme independently proposed by Cohon [33] and Aneja and Nair [4]. However, in this original scheme, they only generate X_{SE1_m}.

The algorithm of phase 1, called `Phase1`, is given in Algorithm 3. This algorithm contains the Procedure 4 called `SolveRecursion`. These algorithms are a generalization of the algorithms given by Przybylski *et al.* for the assignment problem [193].

Algorithm 3 `Phase1`

Parameters $\downarrow$: a biobjective MOCO problem, with $f_1(x)$ and $f_2(x)$ both objectives.
Parameters $\uparrow$: the set X_{SE}.

--| Computation of one lexicographically optimal solution for ($f_1(x)$,$f_2(x)$)
`Solve` $\underset{x\in\mathcal{X}}{\text{lexmin}}\big(f_1(x), f_2(x)\big)$ (1)
Let $\widehat{x}^1$ one optimal solution of (1)

--| Computation of one lexicographically optimal solution for ($f_2(x)$,$f_1(x)$)
`Solve` $\underset{x\in\mathcal{X}}{\text{lexmin}}\big(f_2(x), f_1(x)\big)$ (2)
Let $\widehat{x}^2$ one optimal solution of (2)

--| Computation of X_{SE}
$X_{SE} = \{\widehat{x}^1, \widehat{x}^2\}$
`SolveRecursion`$(\widehat{x}^1 \downarrow, \widehat{x}^2 \downarrow, X_{SE} \updownarrow)$

Procedure 4 `SolveRecursion`

Parameters $\downarrow$: x^r, x^s (with $f_1(x^r) < f_1(x^s)$).
Parameters $\updownarrow$: X_{SE}.

--| Creation of a weight set λ normal to the line segment connecting $f(x^r)$ and $f(x^s)$
$\lambda_1 = f_2(x^r) - f_2(x^s), \lambda_2 = f_1(x^s) - f_1(x^r)$

--| Computation of the set $\mathcal{R}$ of all optimal solutions of the weighted sum problem
`Solve` $\min\limits_{x\in\mathcal{X}}\big(\lambda_1 f_1(x) + \lambda_2 f_2(x)\big)$ (3)
Let $\widehat{f}_\lambda$ the optimal value of (3)
$\mathcal{R} = \big\{x \in \mathcal{X} | \big(\lambda_1 f_1(x) + \lambda_2 f_2(x)\big) = \widehat{f}_\lambda\big\}$

--| Update of X_{SE}
$X_{SE} = X_{SE} \cup \mathcal{R}$

--| Search for new supported efficient solutions
if $\widehat{f}_\lambda < \big(\lambda_1 f_1(x^r) + \lambda_2 f_2(x^r)\big)$ **then**
  Let $x^{t1} \in \text{argmin}\big\{f_1(x) : x \in \mathcal{R}\big\}$
  Let $x^{t2} \in \text{argmin}\big\{f_2(x) : x \in \mathcal{R}\big\}$
  `SolveRecursion`$(x^r \downarrow, x^{t1} \downarrow, X_{SE} \updownarrow)$
  `SolveRecursion`$(x^{t2} \downarrow, x^s \downarrow, X_{SE} \updownarrow)$

As a first stage, the set X_{SE} is initialized with both lexicographic optimal solutions corresponding to $\underset{x\in\mathcal{X}}{\text{lexmin}}\big(f_1(x), f_2(x)\big)$ and $\underset{x\in\mathcal{X}}{\text{lexmin}}\big(f_2(x), f_1(x)\big)$, respectively.

The lexicographic optimal solutions can be obtained by using the Algorithm 1 of chapter 2.

An alternative way to generate the lexicographic optimal solutions, in the case of $p = 2$ and $Y \subset \mathbb{N}^2$, is the following method [193]. In this method, for computing the lexicographic optimal solution $\widehat{x}^1$ corresponding to $\underset{x\in\mathcal{X}}{\text{lexmin}}\big(f_1(x), f_2(x)\big)$, the problem only taking into account the first objective is initially solved; the weight set is thus equal to $(1, 0)$ (the optimal solution of this problem is called $\widehat{x}'^1$). Then, to improve the second objective without degrading the first objective, the single-objective problem with a weight set defined by the normal to the line segment connecting $\big(f_1(\widehat{x}'^1), f_2(\widehat{x}'^1)\big)$ and $\big(f_1(\widehat{x}'^1)+1, -1\big)$, that is $\lambda = \big(f_2(\widehat{x}'^1)+1, 1\big)$, is solved, which allows to obtain the optimal solution of $\underset{x\in\mathcal{X}}{\text{lexmin}}\big(f_1(x), f_2(x)\big)$. The second lexicographic optimal solution $\widehat{x}^2$ corresponding to $\underset{x\in\mathcal{X}}{\text{lexmin}}\big(f_2(x), f_1(x)\big)$ is found in the same way by using $\widehat{x}'^2$, optimal solution of the problem with only the second objective considered, and solving an additional single-objective problem with $\lambda = \big(1, f_1(\widehat{x}'^2) + 1\big)$.

This particular method has the advantage, contrary to the Algorithm 1 of chapter 2, to be based on two weighted sums, and therefore to not modify the structure of the problem.

Secondly, the main part of the algorithm is performed, which is recursively formulated. The `SolveRecursion` procedure takes initially the lexicographic optimal solutions as enters and the set X_{SE} to update.

In this procedure, a weight set λ normal to the line segment connecting $f(x^r)$ and $f(x^s)$ is first defined. We then compute the set $\mathcal{R}$ of *all* optimal solutions of the weighted sum problem defined by the weight set λ. It is indeed necessary to obtain all the optimal solutions since many different non-dominated points can have the same weighted sum objective value.

If we do not enumerate all optimal solutions of the weighted sum problem, as in the original method of Aneja and Nair, we will miss non-extreme supported efficient solutions, and therefore, only X_{SE1_m} is guaranteed to be generated.

Note also that if only a minimal complete set X_{SE_m} is wanted, we still need this enumeration algorithm to generate the non-extreme supported non-dominated points.

Two cases can occur for the set $\mathcal{R}$:

1. $\mathcal{R} \cap \{x^r, x^s\} = \varnothing$ (Figure 3.2). In this case, $\lambda_1 f_1(x) + \lambda_2 f_2(x) < \lambda_1 f_1(x^r) + \lambda_2 f_2(x^r) = \lambda_1 f_1(x^s) + \lambda_2 f_2(x^s) \;\; \forall\, x \in \mathcal{R}$. All solutions of $\mathcal{R}$ are new supported efficient solutions. Let define x^{t1} the solution of $\mathcal{R}$ for which the minimum of $f_1(x)$ is attained and x^{t2} the solution of $\mathcal{R}$ for which the minimum of $f_2(x)$ is attained. Two new weighted sum problems have to be solved with weights corresponding to the normals between $f(x^r)$ and $f(x^{t1})$ as well as $f(x^{t2})$ and $f(x^s)$. This is done by running the `SolveRecursion` procedure twice. In the particular situation where the weighted sum problem has a unique optimal solution, we have $x^{t1} = x^{t2}$.

2. $\{x^r, x^s\} \subset \mathcal{R}$ (Figure 3.3). In this case, $\lambda_1 f_1(x) + \lambda_2 f_2(x) = \lambda_1 f_1(x^r) + \lambda_2 f_2(x^r) = \lambda_1 f_1(x^s) + \lambda_2 f_2(x^s) \;\; \forall\, x \in \mathcal{R}$. Therefore, all solutions of $\mathcal{R}$ except x^r and x^s are new non-extreme supported efficient solutions. The search in this direction stops since the solutions of $\mathcal{R}$ do not allow to define a new weight set.

Phase 2

In phase 2, X_{NE} is generated. Information from phase 1 is used to reduce the search space since for $p = 2$, non-supported efficient solutions are always located, in the objective space, in the interior of the right triangle defined by two consecutive non-dominated supported points.

We will not detail the phase 2 since this phase is problem dependent, in opposition to phase 1 which remains essentially unchanged with the problem considered.

Phase 2 is generally enumerative.

A first technique is to use lower and upper bounds, and to fix some variables, in order to reduce the enumeration in each right triangle.

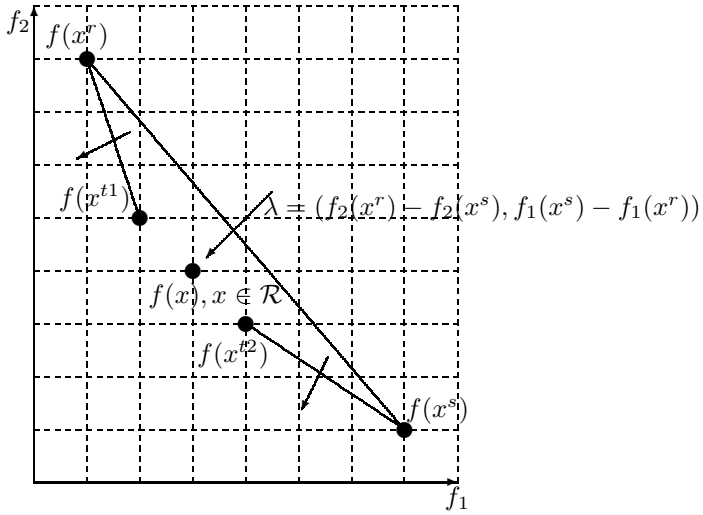

Figure 3.2: Illustration of the functioning of phase 1 of the two-phase method (case 1): three new supported non-dominated points have been found. The search continues by defining two new weight sets: one normal to the segment line connecting $f(x^r)$ and $f(x^{t1})$ and the other one normal to the segment line connecting $f(x^{t2})$ and $f(x^s)$.

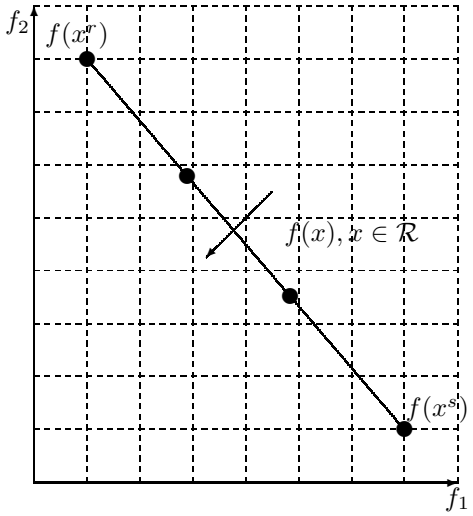

Figure 3.3: Illustration of the functioning of phase 1 of the two-phase method (case 2): two new (non-extreme) supported non-dominated points have been found. However, the search stops in this direction since $\lambda_1 f_1(x) + \lambda_2 f_2(x) = \lambda_1 f_1(x^r) + \lambda_2 f_2(x^r) = \lambda_1 f_1(x^s) + \lambda_2 f_2(x^s) \quad \forall\, x \in \mathcal{R}$.

A second approach, developed recently by Przybylski *et al.* [193], is to use a ranking algorithm to solve the weighted sum problems, that is an algorithm that finds solutions in increasing order of the values given by the weighted sum. In their work concerning the assignment problem, Przybylski *et al.* [193] have shown that this approach is more efficient and simpler to apply than the preceding technique. However, this approach requires a ranking algorithm for the single-objective version of the biobjective problem considered.

Summary

The two-phase method has been applied to a number of problems such as network flow [148, 195], spanning tree [196], assignment [193, 231] and knapsack problems [240].

The drawback of the method is that many enumeration problems have to be solved, even if only a minimal complete set is wanted. Also, ranking algorithms, efficient in phase 2, are not popular in the CO community. On the other hand, the method respects the structure of the problem.

Originally, the method is suited to solve biobjective problems. Indeed, the adaptation of phase 1 to MOPs with $p \geq 3$ is not easy, as pointed out by Solanki *et al.* [211] and Przybyslki *et al.* [194] through two difficulties. The first difficulty emphasized is the determination of the initial points. In biobjective optimization, we need two points to define a weight set normal to the segment line connecting these two points, and there are generally two lexicographic optimal points. On the other hand, in MOPs with $p \geq 3$, p points are needed to define a hyperplane in $\mathbb{R}^p$. But there might be more than p lexicographic optimal points (at most $p!$), that makes unclear which points to choose to define a hyperplane to start the procedure. The second difficulty is that, in opposition to the biobjective case, the normal to a hyperplane defined by p non-dominated points does not necessarily have positive components. Therefore, solving the weighted sum problems with the weight set defined by this normal does not necessarily yield supported non-dominated points (see [194] for an example).

Despite these difficulties, Przybyslki *et al.* [194] and Özpeynirci [180] have generalized the method to MOPs. Both authors use a weight space decomposition, and Özpeynirci introduces a set of dummy points to improve the decomposition. On the other hand, in both methods, only $X_{SE1_{(m)}}$ is generated. The performances of these two methods have not yet been compared.

3.3 Approximation algorithms

Contrary to single-objective CO [10], the interest in approximation methods for MOO is relatively recent. In single optimization, an approximation algorithm A is a (polynomial time) algorithm that finds a feasible solution x^A that is guaranteed to be within a certain ratio of the optimal solution $\widehat{x}$, that is $f(x^A) \leq rf(\widehat{x})$. The constant r (≥ 1) is often called the performance or approximation ratio.

Ehrgott defined in 2000 [55] in a multicriteria context two performance ratios, in the case where the set X_E is approximated by only one single heuristic solution (that allows to use norms, represented by $\|\cdot\| : \mathbb{R}^p \to \mathbb{R}_{\geq}$).

Let $x^A \in \mathcal{X}$ and $x^* \in X_E$.

The performance ratio R_1 of x^A with respect to x^* is

$$R_1(x^A, x^*) := \frac{|\,\|f(x^A)\| - \|f(x^*)\|\,|}{\|f(x^*)\|}$$

An algorithm A that finds a feasible solution of the MOCO problem is an $r_1(n)$-approximation algorithm if

$$R_1(x^A(I), x^*) \leq r_1(|I|)$$

for all instances I of the problem and for all efficient solutions of each instance of the MOCO problem.

The performance ratio R_2 of x^A with respect to x^* is

$$R_2(x^A, x^*) := \frac{\|f(x^A) - f(x^*)\|}{\|f(x^*)\|}$$

An algorithm A that finds a feasible solution of the MOCO problem is an $r_2(n)$-approximation algorithm if

$$R_2(x^A(I), x^*) \leq r_2(|I|)$$

for all instances I of the problem and for all efficient solutions of each instance of the MOCO problem.

An algorithm with performance ratio $r_1 = 1$ can be obtained by solving

$$\min_{x \in \mathcal{X}} \|f(x)\|$$

which can however be a hard problem [55].

These ratios, as they deal with only one heuristic solution, are not very interesting in practice. It is more relevant to define ratios for an approximation composed of a *set* of solutions.

An approximation ratio, very close to the ratio defined for single-objective optimization, is defined as follows [67, 200, 242]:

A set of feasible solutions $X' \subset \mathcal{X}$ is called ϵ-efficient (or ϵ-optimal) if for all $x \in \mathcal{X}$ there is some $x' \in X'$ such that

$$f_k(x') \leq (1 + \epsilon) f_k(x) \quad \forall\, k \in \{1, \ldots, p\}$$

We have thus the same ratio $r = (1 + \epsilon)$ for all objectives.

From this definition, different types of approximation algorithms have been defined [236]:

- A $(1 + \epsilon)$-approximation algorithm A_ϵ is an algorithm that runs in polynomial time and produces an ϵ-efficient solution set for a MOCO problem.
- A polynomial-time approximation scheme (PTAS) is a family of algorithms that contains for each $\epsilon > 0$ a $(1+ \epsilon)$-approximation algorithm A_ϵ.
- A fully polynomial-time approximation scheme (FPTAS) is a PTAS for which A_ϵ is polynomial in ϵ^{-1}.

The literature related to approximation algorithms for MOO is rather scarce:

- In 1979, Hansen [101] gave a FPTAS for the multicriteria shortest path problem. This problem has also been studied by Warburton in 1987 [242].
- In 1990, Ruhe and Fruhwirth [200] presented a $(1 + \epsilon)$-approximation algorithm of the bicriteria network flow problem.
- In 1995, Safer and Orlin [201, 202] obtained results on the existence of FPTAS for some multicriteria network flow, knapsack and scheduling problems.
- In 2000, Papadimitriou and Yannakakis [182] gave an interesting general result. They showed than an ϵ-efficient set can always be built by partitioning the objective space in hyper rectangles such that the ratio of the largest to the smallest coordinate is $(1 + \epsilon)$ for each objective. They also showed that if the single-objective counterpart of the MOCO problem can be solved by a pseudo-polynomial time algorithm (polynomial in the numeric value of the input), then there exists a FPTAS for the MOCO problem.
- In 2008, in his thesis, Özpeynirci [180] proposed a general approximation algorithm to generate a subset of the extreme supported non-dominated points of MOCO problems. The quality of the approximations is measured with lower and upper bound sets of Y_N, which allows to provide a worst case proximity measure.

There are also some works concerning the MOMKP and the MOTSP, that will be reviewed respectively in chapter 4 and chapter 5.

3.4 Metaheuristics

Metaheuristics are general concepts for producing heuristics to solve a large class of computational problems, generally optimization problems. Metaheuristics guide the exploration of the search space such that intensification is realized in promising regions, that are identified by mechanisms of diversification, which allow to explore as much as possible the search space. They are generally stochastic and do not give any guarantee of optimality. However, in practice, metaheuristics are reputed for giving approximations of high quality.

Many different metaheuristics have been developed, from 1975 up to now. Among these methods, the most popular are hill-climbing, simulated annealing (SA) [134], tabu search (TS) [90], genetic algorithm (GA) [104], ant colony optimization (ACO) [48], greedy randomized adaptive search procedure (GRASP) [68], variable neighborhood search (VNS) [175], path-relinking (PR) [86] and iterated local search (ILS) [157].

For a recent survey about these methods, we refer to the books of Teghem and Pirlot [222], Glover and Kochenberger [89], Hoos and Stützle [106] or to the more recent book of Talbi [218].

The metaheuristics have rapidly been adapted to solve MOPs, since the first multiobjective metaheuristic (MOMH) has been developed by Schaffer in 1984 [205], with an adaptation of GAs. But it is mainly since the beginning of the nineties, with the independent works of Serafini [209] and Ulungu [229] in 1992, on the adaptation of SA to MOPs, that a stream of research started and that many different adaptations of metaheuristics to MO problems have been worked out.

The aim of all MOMHs is to generate a good approximation of X_E, that we will call $\widetilde{X}_E$. An approximation $\widetilde{X}_E$ is a set composed of feasible solutions $x \in \mathcal{X}$. The set of all admissible sets $\widetilde{X}_E$ is denoted as ψ. To measure the quality of the approximations, we will need an order on ψ, to be able to say when an approximation is better than another.

From [100], we have the following orders (see Table 3.1) based on Pareto dominance between two approximations $\widetilde{X}_{E_1}$ and $\widetilde{X}_{E_2}$.

Table 3.1: Comparison between two approximations $\widetilde{X}_{E_1}$ and $\widetilde{X}_{E_2}$ based on Pareto dominance.

Strictly dominates	$\widetilde{X}_{E_1} \prec\prec \widetilde{X}_{E_2}$	Every $x_2 \in \widetilde{X}_{E_2}$ is strictly dominated by at least $x_1 \in \widetilde{X}_{E_1}$
Dominates	$\widetilde{X}_{E_1} \prec \widetilde{X}_{E_2}$	Every $x_2 \in \widetilde{X}_{E_2}$ is dominated by at least $x_1 \in \widetilde{X}_{E_1}$
Better	$\widetilde{X}_{E_1} \lhd \widetilde{X}_{E_2}$	Every $x_2 \in \widetilde{X}_{E_2}$ is weakly dominated by at least $x_1 \in \widetilde{X}_{E_1}$ and $\widetilde{X}_{E_1} \neq \widetilde{X}_{E_2}$
Weakly dominates	$\widetilde{X}_{E_1} \preceq \widetilde{X}_{E_2}$	Every $x_2 \in \widetilde{X}_{E_2}$ is weakly dominated by at least $x_1 \in \widetilde{X}_{E_1}$
Incomparable	$\widetilde{X}_{E_1} \| \widetilde{X}_{E_2}$	Neither $\widetilde{X}_{E_1}$ weakly dominates $\widetilde{X}_{E_2}$ nor $\widetilde{X}_{E_2}$ weakly dominates $\widetilde{X}_{E_1}$

We see that two approximations $\widetilde{X}_{E_1}$ and $\widetilde{X}_{E_2}$ can be incomparable, which actually in practice often occurs.

To compare approximations in MOO is thus not easy and this problem will be discussed in the next section.

In all MOMHs, we need to update the approximation $\widetilde{X}_E$, containing all the potentially efficient solutions found so far. For each new solution generated, we have to check if the solution is not dominated by at least one solution of $\widetilde{X}_E$. If it is the case, all solutions of $\widetilde{X}_E$ that are eventually dominated by the new solution have to be removed. Also, as MOMHs generally draw up an approximation of a minimal complete set X_{E_m}, the new solution is not added if there is already an equivalent solution in $\widetilde{X}_E$. The approximation $\widetilde{X}_E$ is always what we called a non-dominated set, which means that all solutions of $\widetilde{X}_E$ are incomparable.

We give in Procedure 5 the procedure `AddSolution` allowing to update the approximation $\widetilde{X}_E$, when a new solution p is added to $\widetilde{X}_E$. This procedure has four parameters, the set $\widetilde{X}_E$ to actualize, the new solution p, its evaluation $f(p)$ and an optional boolean variable called *Added* that returns $True$ if the new solution has been added (this information can be useful for some MOMHs). Every MOMH uses this kind of procedure.

Procedure 5 `AddSolution`

Parameters $\updownarrow$: a set $\widetilde{X}_E$ of potentially efficient solutions.
Parameters $\downarrow$: a solution p, its evaluation $f(p)$.
Parameters $\uparrow$: *Added* (optional).

$Added \leftarrow true$
for all $x \in \widetilde{X}_E$ **do**
  if $f(x) \preceq f(p)$ **then**
    $Added \leftarrow false$
    $Break$ --| We leave the for loop
  if $f(p) \prec f(x)$ **then**
    $\widetilde{X}_E \leftarrow \widetilde{X}_E \setminus \{x\}$
if $Added = true$ **then**
  $\widetilde{X}_E \leftarrow \widetilde{X}_E \cup \{p\}$

It is important to have an efficient implementation of this procedure, since this procedure is called each time a new solution is generated.

A basic implementation is to compare the new solution p to all solutions of $\widetilde{X}_E$ (by using a linear list). However, for biobjective problems, a non-dominated set respects the following propriety: for two solutions x^1 and x^2 belonging to the non-dominated set, $f_1(x^1) < f_1(x^2)$ implies that $f_2(x^1) > f_2(x^2)$. By keeping the linear list sorting according to one of the objectives (that also implies that the list is sorted according to the other objective), and by using this propriety, it is possible to implement a procedure allowing to compare the new solution to only a part of the potentially efficient solutions.

For MOPs with $p \geq 3$, this propriety does not hold anymore and it is not easy to sort the non-dominated set, except by using trees. The quad-tree data structure [215], has been used and studied as structure to store potentially efficient solutions. However, as pointed out by Mostaghim and Teich [177], this structure is only efficient when the number of objectives and number of solutions is not too high (in their study, less than about 3000 potentially efficient solutions). Otherwise, using the linear list is more efficient since the management of the tree becomes very time-consuming.

It is also important to avoid to run the `AddSolution` procedure if the new generated solution can be easily discarded (for instance, dominated by the current solution if the new solution is a neighbor of this solution).

In order to present the MOMHs, we distinguish three groups: local search, evolutionary algorithms and finally hybrid approaches. We start by presenting MOMHs based on local search.

3.4.1 Local search algorithms

In Algorithm 6, we give the general framework of a local search (LS) metaheuristic with the aim of solving a single-objective optimization problem (minimization problem, with $f : \mathbb{R}^n \rightarrow \mathbb{R}$). LS are metaheuristics that use only one solution, at each iteration, to explore the search space with a neighborhood function. A neighborhood function, called $\mathcal{N}(x)$, is a function that generates a set of new feasible solutions, called neighbors, from a unique solution x.

The difference between metaheuristics based on LS mainly depends on the way the neighbor y is selected.

In the hill-climbing method, the neighbor y having a better evaluation than the best solution $\widetilde{x}$ found so far is selected. If there is none, the method stops. In this case, the solution $\widetilde{x}$ is a local optimum which means that this solution cannot be improved with the considered neighborhood.

In SA, the neighborhood is generally not completely generated from the current solution x^c, but stops as soon as a neighbor checks the acceptation rule. A neighbor y is accepted to be the next current solution if it is better than x^c or with a certain probability if it is not better than x^c. The probability rule is based on the analogy with the annealing technique in

Algorithm 6 `LS` (single-objective optimization)

Parameters ↓: an initial solution $x^0 \in \mathcal{X}$, a neighborhood function $\mathcal{N}(x)$.
Parameters ↑: an approximation $\widetilde{x}$ of the optimal solution $\widehat{x}$.

--| Initialization of $\widetilde{x}$ and a current solution x^c with the initial solution x^0
$\widetilde{x} \leftarrow x^0$
$x^c \leftarrow x^0$
repeat
  --| Generation of the neighbors of x^c
  Generate $\mathcal{N}(x^c)$
  --| Selection of a neighbor y
  Select $y \in \mathcal{N}(x^c)$
  if $\big(f(y) < f(\widetilde{x})\big)$ **then**
    $\widetilde{x} \leftarrow y$
  $x^c \leftarrow y$
until stop criterion met

metallurgy [172]. The probability for a neighbor y to be accepted if it is not better than x^c is equal to $e^{(\frac{f(x^c)-f(y)}{T})}$, where T is a parameter that is generally reduced during the algorithm, in order to initially wander towards a broad region of the search space, to find promising regions, for afterwards drifting towards narrower regions.

In TS, the best non-tabu neighbor is selected. A tabu neighbor is a solution that has already been visited during some previous iterations. With this aim, a tabu list of limited size t (the tabu *tenure*) is used to store the previous visited solutions. However, in practice, it is more efficient, in term of implementation, to store the move that allowed to generate the neighbor from the current solution. This implies that sometimes a good solution can be obtained by doing a tabu move, which is however forbidden. To avoid that, an *aspiration* criterion is often added, to allow the realization of a tabu move if the move induces an improvement.

These diversification techniques, included in SA and TS, save the methods from cycling and being stuck in a local optimum, as it happens with the hill-climbing method.

In MOO, we do not need an approximation of a unique solution, but an approximation of a set of solutions. We thus do not work anymore on the search space $\mathcal{X}$ since we are looking for the best approximation $\widetilde{X}_E \in \psi$. Therefore, a LS adapted to solve MOPs (called MOLS) should work with sets, and not with a unique solution. The neighborhood function should also work with sets, that is, from a set, the neighborhood should produce a set of other sets, the neighbors.

Following that, the straightforward adaptation of LS of Algorithm 6 to MOPs gives the Algorithm 7 (called MOLS1 — a second version will be presented after). In this algorithm, the notation $\lhd$ refers to the "Better" order, as defined in Table 3.1.

From this adaptation, we observe two difficulties:

- The neighborhood function could be time-consuming and not easy to define, since from a set, we need to define a function able to produce a set of sets.
- From all the neighbors (all sets), we need to define which set to select for the next iteration. But many sets could be incomparable.

Considering these difficulties, most MOMHs based on LS do not follow this general framework algorithm.

They follow a less straightforward adaptation of LS to MO, but more practical. The algorithm of this adaptation is given in Algorithm 8, and is called MOLS2.

In this adaptation a unique solution x^c explores the search space $\mathcal{X}$, as in single-objective optimization. The neighborhood of this solution is generated, and each neighbor, not weakly

Algorithm 7 `MOLS1`

Parameters ↓: an initial set $X_0 \in \psi$, a neighborhood function $\mathcal{N}(X)$.
Parameters ↑: an approximation $\widetilde{X}_E$ of the efficient set X_E.

--| Initialization of $\widetilde{X}_E$ and a current set X_c with the initial set X_0
$\widetilde{X}_E \leftarrow X_0$
$X_c \leftarrow X_0$
repeat
  --| Generation of the neighbors of X_c
  Generate $\mathcal{N}(X_c)$
  --| Selection of a neighbor Y
  Select $Y \in \mathcal{N}(X_c)$
  if $Y \lhd \widetilde{X}_E$ **then**
    $\widetilde{X}_E \leftarrow Y$
  $X_c \leftarrow Y$
until Stop criterion met

Algorithm 8 `MOLS2`

Parameters ↓: an initial solution $x^0 \in \mathcal{X}$, a neighborhood function $\mathcal{N}(x)$.
Parameters ↑: an approximation $\widetilde{X}_E$ of the efficient set X_E.

--| Initialization of $\widetilde{X}_E$ and a current solution x^c with the initial solution x^0
$\widetilde{X}_E \leftarrow \{x^0\}$
$x^c \leftarrow x^0$
repeat
  --| Generation of the neighbors of x^c
  Generate $\mathcal{N}(x^c)$
  --| Actualization of $\widetilde{X}_E$ with the neighbors y
  for all $y \in \mathcal{N}(x^c)$ **do**
    if $f(x^c) \not\leq f(y)$ **then**
      `AddSolution`$(\widetilde{X}_E \updownarrow, y \downarrow, f(y) \downarrow)$
  --| Selection of a neighbor y
  Select $y \in \mathcal{N}(x^c)$
  $x^c \leftarrow y$
until Stop criterion met

dominated by the current solution x^c, is useful for updating the set $\widetilde{X}_E$. The neighborhood function is thus similar than in the single-objective version of LS. The difference is mainly in the selection of the next solution y in the neighborhood. Indeed, it is not as easy as in single-objective optimization since many neighbors can be incomparable in terms of Pareto dominance, as illustrated in Figure 3.4. In this figure, seven neighbors are generated from x^c and five are incomparable. The selection of the best neighbor should a priori be realized among these five potentially non-dominated neighbors, but new rules have to be defined to refine the selection.

In this section about MOMHs based on LS, we will first present the unique method, to our knowledge, based on the MOLS1 model, the Pareto local search method. We will then briefly present the methods based on the MOLS2 model.

Method based on MOLS1: Pareto local search

The Pareto local search (PLS) method [6, 185] is not the first adaptation of LS to MO, but is however one of the simplest and natural method for solving MOPs.

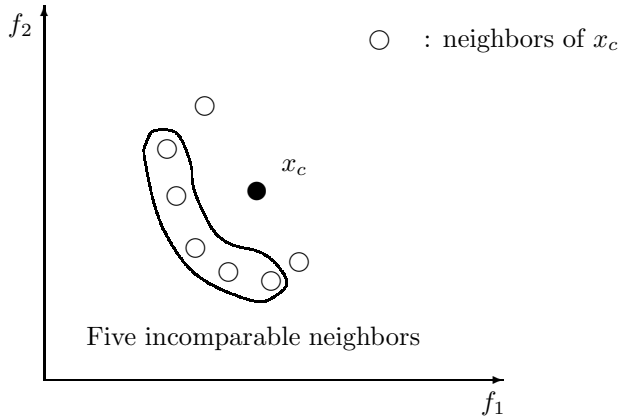

Figure 3.4: Problem of the selection of a neighbor in the MO adaptation of LS: five neighbors are incomparable in term of Pareto dominance.

This method is the generalization in the MO case of the basic hill-climbing metaheuristic. Surprisingly, even if the method is based on the simplest LS, PLS has only been studied in 2002, a long time after the first adaptation of GAs or other more evolved LS like SA or TS.

Paquete *et al.* [185] have exploited this method in the case of the MOTSP but not with an aim of providing results of high quality. The purposes were different: first, to provide reference sets for the assessment of results obtained with more complex algorithms, secondly, to study the notion of Pareto local optimum sets in MO [185, 184], and finally, to study the connectedness between the efficient solutions [61], by identifying cluster of potentially efficient solutions [188].

In 2004, Angel *et al.* [6] studied the PLS method also in the case of the MOTSP. Their version is slightly different than the version of Paquete *et al.*.

In 2006, Basseur [16] used PLS by integrating the method into a GA for solving the MO flow-shop scheduling problem.

The PLS method is based on the MOLS1 model, that is from a set a solutions, a neighborhood is applied to produce new sets (the neighbors). Among these neighbors, one neighbor is selected. If the selected neighbor is better than the best set found so far, this neighbor becomes at the same time the new current set and the best set. The method simply stops when none neighbor is better than the best set. This is thus a straightforward adaptation of the hill-climbing LS to the MO case, and at the end, a local optimum is found. The method does not require any objectives aggregation nor any numerical parameters.

We first present some definitions related to the notion of local optimum in the MO case. Afterwards, we present the two different versions of the PLS method (Paquete *et al.* [185] and Angel *et al.* [6]), both methods being a simplification of the MOLS1 model. Details are given since both versions of PLS will be used later in this work.

We start with the definition of a *Pareto local optimum*, which is a generalization, in the MO case, of the concept of local optimum.

We consider a solution $p \in \mathcal{X}$ and $\mathcal{N}$ a neighborhood of p

Definition 23 *Pareto local optimum [184]: we say that a solution p is a* Pareto local optimum *with respect to (w.r.t.) $\mathcal{N}$ if, and only if, there is no solution $p' \in \mathcal{N}(p)$ such that $f(p') \prec f(p)$.*

As in MOO, we are looking for a set of solutions, we define the notion of *Pareto local optimum set*. We consider a set $P \in \psi$.

Definition 24 *Pareto local optimum set [184]: we say that P is a* Pareto local optimum set *w.r.t. $\mathcal{N}$ if, and only if, P contains only Pareto local optimum solutions w.r.t. $\mathcal{N}$.*

But, as this set may contain dominated solutions, we define the notion of *non-dominated Pareto local optimum set*.

Definition 25 *Non-dominated Pareto local optimum set [183]: we say that P is a* non-dominated Pareto local optimum set *w.r.t. $\mathcal{N}$ if, and only if, it is a Pareto local optimum set w.r.t. $\mathcal{N}$ and it is a non-dominated set, that is:*

$$\forall\, p \in P, \nexists\, p' \in P |\, f(p') \prec f(p)$$

Also, as a non-dominated Pareto local optimum set w.r.t. $\mathcal{N}$ can still be improved by adding solutions $p' \in \mathcal{N}(p)$ such that $f(p) \not\prec f(p')$, we define the notion of *critical non-dominated Pareto local optimum set.*

Definition 26 *Critical non-dominated Pareto local optimum set: we say that P is a* critical non-dominated Pareto local optimum set *w.r.t. $\mathcal{N}$ if, and only if, it is a non-dominated Pareto local optimum set and*

$$\forall\, p \in P, \forall\, p' \in (\mathcal{N}(p) \backslash P), \exists\, p'' \in P \,|\, f(p'') \prec f(p')$$

With this definition, if for a set the following condition is met:

$$\exists\, p \in P \,|\, \exists\, p' \in (\mathcal{N}(p) \backslash P) \text{ with } f(p) = f(p'),$$

the set is not a critical non-dominated Pareto local optimum set since we cannot find a solution p'' belonging to P that dominates the solution p'. Therefore, all equivalent solutions of the solutions of the set have to be in the set to be a critical non-dominated Pareto local optimum set.

Finally, we define the notion of *minimal critical non-dominated Pareto local optimum set* as follows:

$$\forall\, p \in P, \forall\, p' \in \mathcal{N}(p), \exists\, p'' \in P \,|\, f(p'') \preceq f(p') \text{ and } \forall p \in P, \nexists p' \in P \,|\, f(p) = f(p')$$

Therefore, such a set does not contain any equivalent solutions.

Note that Paquete *et al.* [186] defined the notion of *maximal Pareto local optimum set* as follows:

$$\forall\, p \in P, \forall\, p' \in \mathcal{N}(p), \exists\, p'' \in P \,|\, f(p'') \preceq f(p')$$

which is a little bit ambiguous, since it is only a necessary condition to be a minimal critical non-dominated Pareto local optimum set.

It is needless to say that the efficient set X_E is a critical non-dominated Pareto local optimum set w.r.t. any neighborhood $\mathcal{N}$, because if it was not the case, the set X_E could be improved by adding a non-dominated neighbor.

We present now the two different versions of PLS, that will be experimented in this work (see chapters 7, 8 and 9). Both versions need an initial set; that can be composed of a single solution or of a list of potentially efficient solutions. However, Paquete *et al.* and Angel *et al.* used both a single solution. Both versions return a minimal critical non-dominated Pareto local optimum set.

- **PLS1**

  We begin with the description of the version of Angel *et al.* [6]. In their work, they call the method BLS for biobjective LS. However, for simplicity of notations, we will call it PLS1.

  The method follows the Algorithm 7 of the MOLS1 model, except that only one neighbor set is produced from the current set. Therefore, the selection of the best neighbor does not pose a problem. The neighbor is generated as follows: the neighborhood of each solution of the current set is explored. Each new potentially efficient solution as well as the previous solutions that are not dominated by any neighbors compose the new current set. In this way, the neighbor is always better than the current set, except if no new

Algorithm 9 `PLS1`

Parameters $\downarrow$: an initial population P_0, a neighborhood function $\mathcal{N}(x)$.
Parameters $\uparrow$: an approximation $\widetilde{X}_E$ of the efficient set X_E.

--| Initialization of $\widetilde{X}_E$ and a population P with the initial population P_0
$\widetilde{X}_E \leftarrow P_0$
$P \leftarrow P_0$
--| Initialization of an auxiliary population P_a
$P_a \leftarrow \varnothing$
while $P \neq \varnothing$ **do**
  --| Generation of all neighbors p' of each solution $p \in P$
  for all $p \in P$ **do**
    for all $p' \in \mathcal{N}(p)$ **do**
      if $f(p) \not\preceq f(p')$ **then**
        `AddSolution`($\widetilde{X}_E \updownarrow$, $p' \downarrow$, $f(p') \downarrow$, *Added* $\uparrow$)
        if *Added* $=$ *true* **then**
          `AddSolution`($P_a \updownarrow$, $p' \downarrow$, $f(p') \downarrow$)
  --| P is composed of the new potentially efficient solutions
  $P \leftarrow P_a$
  --| Reinitialization of P_a
  $P_a \leftarrow \varnothing$

potentially efficient solution has been generated. In this case, the method stops since a minimal critical non-dominated Pareto local optimum set has been found (we cannot improved this set with the neighborhood).

The pseudo-code of PLS1 is given in Algorithm 9.

To avoid to explore the neighborhood of solutions that have already been visited, two populations of solutions are used: a population P composed of the solutions for which the neighborhood has to be explored and an auxiliary population P_a that contains the new potentially efficient solutions. When the neighborhood of all solutions of P has been explored, P_a becomes the new population P.

More precisely, the method starts with the population P composed of the potentially efficient solutions given by the initial population P_0. Then, all the neighbors p' of each solution p of P are generated. If a neighbor p' is not weakly dominated by the current solution p, we try to add the solution p' to the approximation $\widetilde{X}_E$ of the efficient set, which is updated with the procedure `AddSolution` (Procedure 5). If the solution p' has been added to $\widetilde{X}_E$, the boolean variable *Added* is true and the solution p' is added to P_a, which is also updated with the procedure `AddSolution`. Therefore, P_a is only composed of (new) potentially efficient solutions. Once all the neighbors of each solution of P have been generated, the algorithm starts again, with P equal to P_a, until $P = P_a = \varnothing$. The auxiliary population P_a is used such that the neighborhood of each solution of the population P is explored, even if some solutions of P become dominated following the addition of a new solution to P_a. Thus, sometimes, in this version, neighbors are generated from a dominated solution.

- **PLS2**

  We now present the PLS method of Paquete *et al.* [185]. In this version, called PLS2, the population P is immediately updated after the addition of a neighbor. Therefore, in opposition to PLS1, the neighbors are never generated from a dominated solution. The main difference in the pseudo-code of PLS2, given in Algorithm 10, compared to that of PLS1, is that no auxiliary population is used.

We only have to employ a boolean variable *Deleted* to check if the current solution p has been eliminated from the population P following the addition of a neighbor p' of p. If it is not the case, the solution p has to be removed from P as the neighborhood of p has already been explored.

Consequently, the results given by PLS2 depend on the order according to which the solutions of P are examined, which was not the case with PLS1.

Algorithm 10 `PLS2`

Parameters $\downarrow$: an initial population P_0, a neighborhood function $\mathcal{N}(x)$.
Parameters $\uparrow$: an approximation $\widetilde{X}_E$ of the efficient set X_E.

--| Initialization of $\widetilde{X}_E$ and a population P with the initial population P_0
$\widetilde{X}_E \leftarrow P_0$
$P \leftarrow P_0$
while $P \neq \varnothing$ **do**
  --| Generation of all neighbors p' of each solution $p \in P$
  for all $p \in P$ **do**
    $Deleted \leftarrow false$
    for all $p' \in \mathcal{N}(p)$ **do**
      if $f(p) \not\preceq f(p')$ **then**
        `AddSolution`($\widetilde{X}_E \updownarrow$, $p' \downarrow$, $f(p') \downarrow$, $Added \uparrow$)
        if $Added = true$ **then**
          `AddSolution`($P \updownarrow$, $p' \downarrow$, $f(p') \downarrow$)
      if $f(p') \prec f(p)$ **then**
        $Deleted \leftarrow true$
  --| We remove p from P if p has not already been removed
  if $Deleted = false$ **then**
    $P \leftarrow P \setminus \{p\}$

Methods based on MOLS2

In this section, we briefly review the MOMHs based on the MOLS2 model, that is the neighborhood of a single solution is explored and each new potentially efficient neighbor found is added to $\widetilde{X}_E$. We first present three methods based on SA and then two methods based on TS.

- **SA**

  In SA, a neighbor is generated from the current solution, and is eventually considered as the next current solution, depending on the rule of acceptation.

  To adapt SA to MO, we need to define this rule of acceptation in a MO context. Three cases are distinguished, as illustrated in Figure 3.5 for a biobjective problem. We consider a move from the current solution x^c to a neighbor y:

  - Case a : $\Delta f_k(y) = f_k(y) - f_k(x^c) \leq 0 \quad k = 1, \ldots, p$: improving move.
  - Case b : $\Delta f_k(y) \geq 0 \quad k = 1, \ldots, p$ (with at least one strict equality): degrading move.
  - Case c : $\exists\, k, k'$ such that $\Delta f_k(y) < 0$ and $\Delta f_{k'}(y) > 0$: non-dominated move.

  An improving move will be accepted with a unary probability. We now have to define the probabilities to accept a degrading move or a non-dominated move.

  The adaptations of SA to MO mainly differ on this point.

  We present three MO methods based on SA: the MOSA method developed by Ulungu and Teghem [229], the SMOSA method developed by Serafini [209] and the PSA method developed by Czyzak and Jaszkiewicz [35].

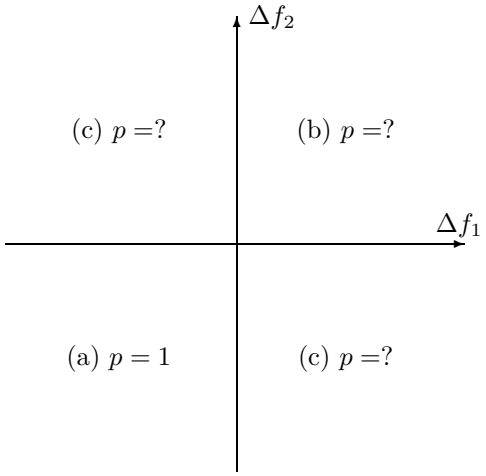

Figure 3.5: Representation of the three possible moves in the biobjective case.

– **MOSA**

The strategy proposed by Ulungu and Teghem [229] to deal with the cases b and c is to used a scalarization function $s(f(x), \lambda)$. The variation $\Delta s(y)$ of a move from x^c to y is equal to:

$$\Delta s(y) = s\big(f(y), \lambda\big) - s\big(f(x^c), \lambda\big)$$

Any scalarizing functions can be used. In MOSA, the weighted sum (see section 3.2.1) is selected.

The value of $\Delta s(y)$ is then used to determine the probability of acceptation of a move:

* $\Delta s(y) \leq 0$: the move will be accepted with a unary probability, as in the case a.
* $\Delta s(y) > 0$: the move will be accepted with a probability equal to $e^{\frac{-\Delta s(y)}{T}}$, with T the temperature parameter of SA.

If the aim is to generate a diversified approximation $\widetilde{X}_E$ (solutions in every region of the objective space), MOSA has to be applied several times with a well-diversified set of weight sets, since each weight set λ gives priority to a direction in the objective space. Once MOSA has been applied for each weight set λ, the different approximations $\widehat{X}_{E_\lambda}$ obtained for each weight set λ are merged to achieve a unique approximation $\widehat{X}_E$.

The MOSA method has been applied to a large number of MOCO problems: knapsack problems [232], assignment problems [227], scheduling problems [156] and the vehicle routing problem [226]. An interactive version of MOSA has also been developed and tested on a real situation (problem of homogeneous grouping of nuclear fuel [234]) and with a fictitious decision maker for the knapsack and the assignment problems [223].

– **SMOSA**

The SMOSA method developed independently by Serafini in 1992 [209] is nearly the same as the MOSA method of Ulungu and Teghem, except that at each iteration the weight vector used in the acceptance rule is randomly modified.

– **PSA**

The PSA method developed by Czyzak and Jaszkiewicz in 1998 [35] is a little bit more complex than the two previous ones. But, again, the difference is in the way the weight sets are managed. In PSA, an initial set of solutions is generated. Weight sets are associated to each of these solutions, that are optimized in the same way as

in MOSA. For a given solution belonging to the initial set, the weight set is changed in order to induce a repulsion mechanism assuring dispersion of the solutions over all regions of the Pareto front.

The PSA method has been applied to a large number of problems, from knapsack problems to fuzzy MOCO problems. We refer to [117] for a panel of the applications of PSA.

In [115], Jaszkiewicz compares the results obtained by the two aforementioned SA methods (MOSA and SMOSA) with PSA on the biobjective set covering problem. He shows that MOSA and PSA obtain similar results, while the results of SMOSA are of lower quality. In [116], he compares again these three methods for the biobjective MKP. But here he shows that PSA is the best on most of the instances.

- **TS**

  In TS, neighbors are generated from the current solution, and one neighbor is selected among the neighbors to be the next current solution. The different adaptations of TS to MO mainly differs on this point.

  - **MOTS**

    The first adaptation of TS to MO is due to Gandibleux *et al.* in 1997 [77]. To select the best neighbor, they use the augmented weighted Tchebycheff scalarizing function (see section 3.2.1). The weight set is dynamically updated such that the search process is oriented towards a region of the objective space where few non-dominated points have been generated. A tabu list in the search space is defined to avoid cycling. In [77], the method is applied to an unconstrained permutation problem while in [75], the method is applied to the biobjective KP.

  - **TAMOCO**

    In 2000, Hansen [99] proposed a relatively simple adaptation of TS. The selection of the best neighbor is based on the weighted sum. The particularity of the method is that a set of solutions searching for potentially efficient solutions is managed in parallel. The weight sets are dynamically updated. The method has been applied to the project scheduling problem in 2000 by Viana and Pinho de Sousa [237].

  - **MOTAS**

    In 2000, Loukil *et al.* [155] proposed the MOTAS method. In this adaptation of TS, the selection of the best neighbor is based on weighted Tchebycheff scalarizing function. As in MOSA, the method is applied several times with a well-diversified set of weight sets. They applied the method to several biobjective scheduling problems, by considering different associations of criteria.

  - **TS+HW**

    This TS has been developed by Barichard and Hao in 2002 [15]. This is a classic TS with a Tchebycheff scalarizing function to select the best neighbor, except that an interesting way to measure the diversity has been added. If the diversity measure falls below a certain threshold, a diversification phase is realized: the method is run from a new solution, that is a deteriorated version of the current solution. The tabu values of the moves that have been frequently realized during the last intensification phase is increased. To measure the diversity, they compute the Hamming distance of the last configurations accepted by the algorithm, taken in pairs.

    They applied the method to the MOMKP. Unfortunately, nothing is said about the values of the parameters of the method and, worse, they do not talk about the strategy that they have adopted to fix the weight sets used in the evaluation of the neighbors. This kind of lack makes the method difficult to adapt to other MOCO problems.

3.4.2 Evolutionary algorithms

Evolutionary algorithms (EAs) are generally composed of three fundamental elements:

- A population made up of individuals (also called the *chromosomes*) representing, via a coding, the feasible solutions.
- A fitness function that evaluates the adaptation of the individuals to the environment.
- An evolution process that makes evolve the population by generating and eliminating individuals.

Among the EAs, we will only deal with the genetic algorithms (GAs) [89, 91, 104]. We give the description of the functioning of GAs in Algorithm 11 for a single-objective problem. In this algorithm, we consider a natural coding of the solutions and a crossover operator that only needs two parents to create a single offspring (the standard name given to the result of the crossing). In GAs, a population is evolving through selection, reproduction and mutation operators, in order to obtain a well-diversified population and containing at least one solution of good quality. The individuals of the population are evaluated with a fitness function which is often based on the objective to optimize.

Algorithm 11 `GA` (single-objective optimization)

Parameters $\downarrow$: an initial population P_0 of size nP composed of solutions x in $\mathcal{X}$ (we suppose a natural coding of the solutions), a fitness function $F(x)$, a procedure $SP1(P)$ of selection of the first parent, a procedure $SP2(P)$ of selection of the second parent, a crossover operator $CO(x_1, x_2)$, a mutation operator $M(x)$, the number nP' of reproduction, a selection procedure $S(P)$.
Parameters $\uparrow$: an approximation $\widetilde{x}$ of the optimal solution $\widehat{x}$.

--| Initialization of the population P
$P \leftarrow P_0$
repeat
  --| Generation of a new population P' from P
  $P' \leftarrow \varnothing$
  for $i = 1$ to nP' **do**
    --| Selection of the first parent
    $x^1 \leftarrow SP1(P)$
    --| Selection of the second parent
    $x^2 \leftarrow SP2(P)$
    --| Reproduction between both parents through the crossover operator
    $x^3 \leftarrow CO(x^1, x^2)$
    --| The offspring x^3 is eventually mutated
    $x^4 \leftarrow M(x^3)$
    --| x^4 is added to the new population P'
    $P' \leftarrow P' + \{x^4\}$
  --| Selection of nP individuals in $\{P \cup P'\}$ in order to update the population P
  $P \leftarrow S(\{P \cup P'\})$
until stop criterion met
$\widetilde{x} \leftarrow$ best solution in P

The selection of the parents and the selection of the solutions composing the new population are often realized on the base of the fitness function $F(x)$. Most of the time, the procedure S simply consists in selecting the nP best individuals in $\{P \cup P'\}$.

In single-objective optimization, using a population of solutions can be a little contradictory to the fact that only one good solution is sought. Therefore, in single-objective optimization, LS methods are at least as popular and effective as GAs.

This is not the case in MO. For instance, the overview of Jones *et al.* in 2002 [126] indicates that 70% of the papers about MOPs utilize GAs as primary metaheuristic technique, 24% SA, and 6% TS. Indeed, in GAs, the whole population contributes to the evolutionary process and ideally represents the different interesting regions of the search space where efficient solutions are located. This characteristic makes GAs very attractive for solving MOPs.

The adaptation of GAs to MO (MOGA) is also more natural than with LS. Indeed, since a population of solutions is already managed, it would be enough to replace the population P by a non-dominated set, that is a set containing only potentially efficient solutions. However, this option could be too elitist and, generally, two populations are managed: the population P and the elitist set, that is the approximation set $\widetilde{X}_E$ containing the best potentially efficient solutions found so far. That gives the algorithm presented in Algorithm 12, very similar to the single-objective version except that the approximation set $\widehat{X}_E$ is used and plays a role for the selection. Such an algorithm where the set of potentially efficient sets interferes in the selection is called an *elitist* MOGA. The importance of elite solutions to improve MOGAs has been studied by Laumanns *et al.* in 2001 [147]. He showed that the use of elitism must be accompanied by a strong rate of mutation in order to avoid a premature convergence of the algorithm.

Algorithm 12 `MOGA`

Parameters $\downarrow$: an initial population P_0 of size nP composed of solutions x in $\mathcal{X}$ (we suppose a natural coding of the solutions), a fitness function $F(x)$, a procedure $SP1(P)$ of selection of the first parent, a procedure $SP2(P)$ of selection of the second parent, a crossover operator $CO(x_1, x_2)$, a mutation operator $M(x)$, the number nP' of reproduction, a selection procedure $S(P)$.
Parameters $\uparrow$: an approximation $\widetilde{X}_E$ of the efficient set X_E.

```
--| Initialization of the population P
P ← P_0
--| Initialization of X̃_E
for all x^i ∈ P do
  AddSolution(X̃_E ↕, x^i ↓, f(x^i) ↓)
repeat
  --| Generation of a new population P' from P and X̃_E
  P' ← ∅
  for i = 1 to nP' do
    --| Selection of the first parent
    x^1 ← SP1({P ∪ X̃_E})
    --| Selection of the second parent
    x^2 ← SP2({P ∪ X̃_E})
    --| Reproduction between both parents through the crossover operator
    x^3 ← CO(x^1, x^2)
    AddSolution(X̃_E ↕, x^3 ↓, f(x^3) ↓)
    --| The offspring x^3 is eventually mutated
    x^4 ← M(x^3)
    AddSolution(X̃_E ↕, x^4 ↓, f(x^4) ↓)
    --| x^4 is added to the new population P'
    P' ← P' + {x^4}
  --| Selection between {P ∪ P' ∪ X̃_E} to update the population P
  P ← S({P ∪ P' ∪ X̃_E})
until stop criterion met
```

Another alternative to this algorithm would be to work on the space ψ of the admissible sets, that is to use a population of sets in the place of a population of solutions, but to our knowledge, no MOGA method is based on such a scheme.

We will now present some of the algorithms based on the MOGA model. There are plenty of them and we refer to [32, 42, 126, 151] for a broader overview. In [151], Liefooghe *et al.* do more than presenting MOGAs: they give a general model, in order to easily implement MOMHs through a software called ParadisEO.

The main differences between all MOGAs is in the selection processes (how the parents are selected and how the population is updated). The selection can be based on the performances for one of the objectives [205], on scalarizing functions [178] or on dominance relations of Pareto [73]. It is this last idea that is the most common approach and that has led to several Pareto-based fitness assignment schemes.

A well-known fitness assignment scheme to evaluate the individuals has been introduced (but not exploited) by Goldberg in 1989 [91]. It consists in assigning ranks to each individual of the population.

The illustration of this rank is given in Figure 3.6. In a first stage, all non-dominated individuals of the population are assigned a rank equal to 1 (Front 1 in the figure) and are temporally withdrawn from the population. Then, the new non-dominated individuals take a rank equal to 2 and are in their turn withdrawn from the population (Front 2 in the figure). The process is reiterated as long as there are still individuals in the population. The fitness of an individual is equal to its rank. Therefore, the fitness value of an individual does only depend on itself, but also on the other individuals in the population.

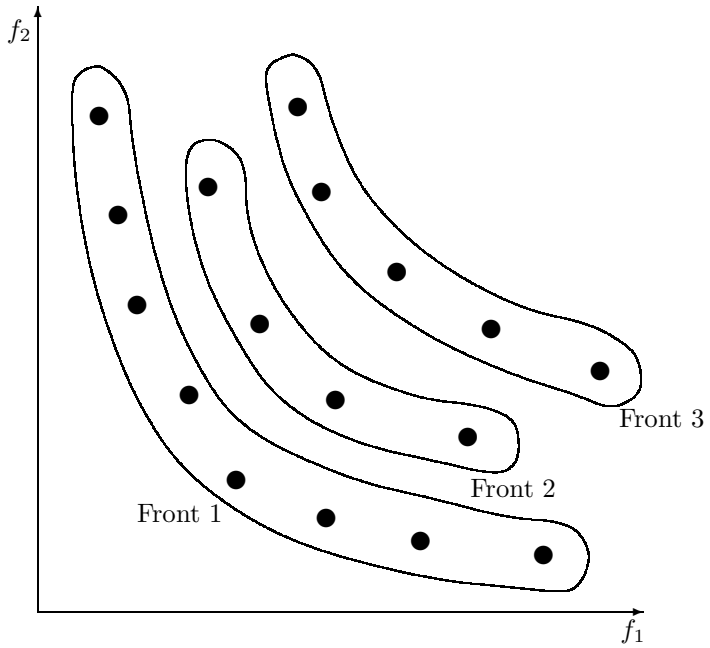

Figure 3.6: Illustration of the functioning of the rank developed by Goldberg [91].

Using ranks based on Pareto dominance relations allows to guide the search towards the efficient set. However, it is still necessary to maintain a diversified population in order to avoid premature convergence and to find a diversified set of potentially non-dominated points.

With this aim, fitness *sharing* is used. Fitness sharing consists in integrating diversity information into the fitness value of an individual. Indeed, in order to avoid a too high number of individuals located in the same region, the fitness of an individual should be penalized according to the number of individuals present in the neighborhood of the individual. The most frequently sharing technique used is based on the notion of *niches*. A niche is a hypersphere centered on individuals and with a radius σ_{sh}. The individuals that are outside of the hypersphere are not considered in the fitness sharing. The fitness function $F(i)$ of an individual i is then defined in the following way:

$$F(i) = \frac{F'(i)}{\sum_{j \in P} \phi\big(d(i,j)\big)}$$

where $F'(i)$ is the value given by the rank based on Pareto dominance and $\phi(d(i,j))$ is the sharing function, computed as follows:

$$\phi\big(d(i,j)\big) = \begin{cases} 1 - \left(\frac{d(i,j)}{\sigma_{sh}}\right)^{\alpha} & \text{if } d(i,j) < \sigma_{sh} \\ 0 & \text{otherwise.} \end{cases}$$

where the parameter α allows to amplify ($\alpha > 1$) or to attenuate ($\alpha < 1$) the sharing function $\phi\big(d(i,j)\big)$ and where $d(i,j)$ is the distance between the individuals i and j that can be defined in the decision space (by using, for instance, the Hamming distance) or in the objective space (by using, for instance, the Euclidean distance).

Among the many adaptations of GAs to MO, we present briefly two popular methods: the strengh Pareto evolutionary algorithm (SPEA2), and the non-dominated sorting genetic algorithm (NSGA-II).

- **SPEA2**

  The SPEA2 method is an elitist MOGA that has been developed by Zitzler *et al.* in 2001 [251]. It is an improved version of SPEA developed by the Zitzler and Thiele in 1998 [253]. In SPEA2, the rank of an individual takes into account the number of individuals that dominate it and the number of individuals dominated by the individual. Clustering in the objective space is used in order to reduce the number of individuals and allows to obtain only one representative individual in small regions of the objective space. The method has originally been applied to the MOMKP.

- **NSGA-II**

  The NSGA-II method has been developed by Deb *et al.* in 2002 [46] and is an improved version of NSGA propsed by Srinivas and Deb in 1994 [212]. In NSGA, the fitness is based on the Goldberg rank. No elitism is introduced in the selection. However, in NSGA-II, the elitist population is used in the selection. A diversity measure is introduced with the notion of *crowding*. The crowding distance of an individual is related to the value of the perimeter of the hypercube having as vertices the points the closest to the individual for each objective. The advantage of this measure is that no parameter is needed. The NSGA-II method has been applied to a large variety of problems, see [43] for an overview.

Most of the MOGAs have been tested on unconstrained non-linear MOPs and are very popular in the engineering community for solving problems in mechanical design or electronics. On the other hand, MOGAs have been poorly considered to solve MOCO problems. Indeed, for dealing with MOCO problems, hybrid approaches seem more efficient. This is the subject of the next section.

3.4.3 Hybrid approaches

Hybrid approaches are nowadays the most competitive metaheuristics based methods to solve optimization problems [22]. By combining the qualities of different metaheuristics, they allow to obtain very good results, often superior to the original methods from which they are based on.

For example, for MOPs, a weak point in LS is that only one solution is evolving in the search space, and it is often needed to restart the search from another point, as done for example in MOSA, by applying the search in different regions with the help of weight sets. On the other hand, GAs have this ability to explore different regions of the search space at the same time since a population, ideally well-diversified, is used. However, GAs do not have generally as good intensification properties as LS. Therefore, a natural method, hybriding GAs and LS was born from these observations.

We only focus our study of the hybrid approaches, on the hybridization between LS an GAs. The common name given to the algorithm resulting from this hybridization is *memetic* algorithm. For other hybrid schemes, we refer the reader to [58].

A memetic algorithm (MA) (also sometimes called genetic local search) is a genetic algorithm where the mutation operator is replaced by a local search applied to every new offspring generated [176]. MAs are particularly well adapted to solve MOPs since a diversified set (from where the interest to use a population) and close to the Pareto front (what is ensured by the LS) is required.

We give in Algorithm 13 the functioning of MOMA, adaptation of MAs to MO. Comparing to MOGA, the mutation operator is replaced by a LS. The LS starts from one solution, the offspring resulting from the crossover operator. We assume that the LS produces only one new solution x_4.

Algorithm 13 `MOMA`

Parameters $\downarrow$: an initial population P_0 of size nP composed of solutions x in $\mathcal{X}$ (we suppose a natural coding of the solutions), a fitness function $F(x)$, a procedure $SP1(P)$ of selection of the first parent, a procedure $SP2(P)$ of selection of the second parent, a crossover operator $CO(x_1, x_2)$, a local search $LS(x)$, the number nP' of reproduction, a selection procedure $S(P)$.
Parameters $\uparrow$: an approximation $\widetilde{X}_E$ of the efficient set X_E.

--| Initialization of the population P
$P \leftarrow P_0$
--| Initialization of $\widetilde{X}_E$
for all $x^i \in P$ **do**
  `AddSolution`$(\widetilde{X}_E \updownarrow, x^i \downarrow, f(x^i) \downarrow)$
repeat
  --| Generation of a new population P' from P and $\widetilde{X}_E$
  $P' \leftarrow \varnothing$
  for $i = 1$ to nP' **do**
    --| Selection of the first parent
    $x^1 \leftarrow SP1(\{P \cup \widetilde{X}_E\})$
    --| Selection of the second parent
    $x^2 \leftarrow SP2(\{P \cup \widetilde{X}_E\})$
    --| Reproduction between both parents through the crossover operator
    $x^3 \leftarrow CO(x^1, x^2)$
    `AddSolution`$(\widetilde{X}_E \updownarrow, x^3 \downarrow, f(x^3) \downarrow)$
    --| The LS starts from the offspring x^3
    $x^4 \leftarrow LS(x^3)$
    `AddSolution`$(\widetilde{X}_E \updownarrow, x^4 \downarrow, f(x^4) \downarrow)$
    --| x^4 is added to the new population P'
    $P' \leftarrow P' + \{x^4\}$
  --| Selection between $\{P \cup P' \cup \widetilde{X}_E\}$ to update the population P
  $P \leftarrow S(\{P \cup P' \cup \widetilde{X}_E\})$
until stop criterion met

In an interesting survey about MAs for MO [139], Knowles and Corne have distinguished three principal groups of authors who have developed MOMAs: Ishibuchi and Murata with IMMOGLS (Ishibuchi Murata multiobjective genetic local search) [110], Jaszkiewicz with two methods, MOGLS (multiobjective genetic local search) [112] and PMA (Pareto memetic algorithm) [115] and finally, Knowles and Corne with M-PAES (memetic Pareto archived evolution strategy) [138]. All these methods are based on the MOMA model.

We briefly present below each of these methods. The three methods IMMOGLS, MOGLS

and PMA are very similar and all based on a scalarizing function with weight sets randomly generated in order to select both parents. The only two points on which the three algorithms differ is how the scalarizing function is used to select both parents for recombination and how the new population is built.

- **IMMOGLS**

  The IMMOGLS method has been developed by Ishibuchi and Murata in 1996 [110]. In this method, the parents are selected from the current population by using the roulette wheel selection scheme based on the value given by the scalarazing function. The new population replaces the preceding one, but some elite solutions are preserved. The method has been applied to the multiobjective flow-shop scheduling problem. Three criteria are considered: makespan, total tardiness and total flowtime.

- **MOGLS**

  The MOGLS method has been developed by Jaszkiewicz in 1998 [112]. In MOGLS, the best solutions in P for the scalarizing function form a temporary population TP of small size. Both parents are randomly select in TP. If the solution x^4 is better than the worst solution in TP (according to the scalarizing function) and different in the decision space than all solutions in TP, the solution x^4 is added to P and to $\widetilde{X}_E$ (if x^4 is potentially efficient). The method has been applied to the MOMKP [114], to the biobjective set covering problem [115] and to the MOTSP [118].

- **PMA**

  In PMA, the selection is based on a tournament. The two parents selected are the winners of a tournament between solutions coming from a sample of size T randomly drawn from the population. The size of T is set in order to guarantee that this selection procedure gives the same quality of offsprings than with the selection procedure of MOGLS. Indeed, the PMA method has been developed by Jaszkiewicz in order to reduce the running time of the selection process of its MOGLS algorithm, while keeping same quality results. If the solution x^4 is better than the second best solution in T (according to the scalarizing function), the solution x^4 is added to P and to $\widetilde{X}_E$ (if x^4 is potentially efficient). The method has been applied to the biobjective set covering problem [115] and the MOTSP [121].

- **M-PAES**

  The M-PAES method has been developed by Knowles and Corne in 2000 [138]. The method is rather different from the three preceding ones, since none scalarizing function is used, either in the LS or in the parents selection. The fitness function of an individual is instead based on Pareto dominance. The LS is the (1+1)-PAES method [136], which is a procedure for maintaining a finite size list of potentially efficient solutions. In (1+1)-PAES, a new solution is generated from the current solution, and, to check if this new solution is accepted, a comparison with the current solution and the population is realized. The rule of acceptation is as follows:

  - If the new solution is dominated by the current solution: not accepted.
  - If the new solution dominates at least one solution of the population: accepted.
  - If the new solution is not dominated by at least one solution of the population but does not dominate any solution of the population: the solution is accepted on the condition that the solution brings diversity. The diversity is measured by a hypergrid created in the objective space.

  In addition, the M-PAES method employs periodically a crossover operator to recombine the solutions generated by the (1+1)-PAES procedure.

The M-PAES method is known to be parsimonious in term of number of evaluations in comparison with MOMAs based on scalarizing functions, since in these algorithms, solutions are rejected if they are not good on the current scalarizing function, even if the solutions could be potentially efficient. In M-PAES, as Pareto dominance is used, no potentially efficient solution is discarded.

M-PAES has been applied to the MOMKP [138].

3.5 Performance assessment of MOMHs

The assessment of the performances of MOMHs is a hot topic, and still under research [140, 252, 255]. The main reason is that, in term of Pareto dominance, the approximations returned by MOMHs are most of the time incomparable, as illustrated in Figure 3.7.

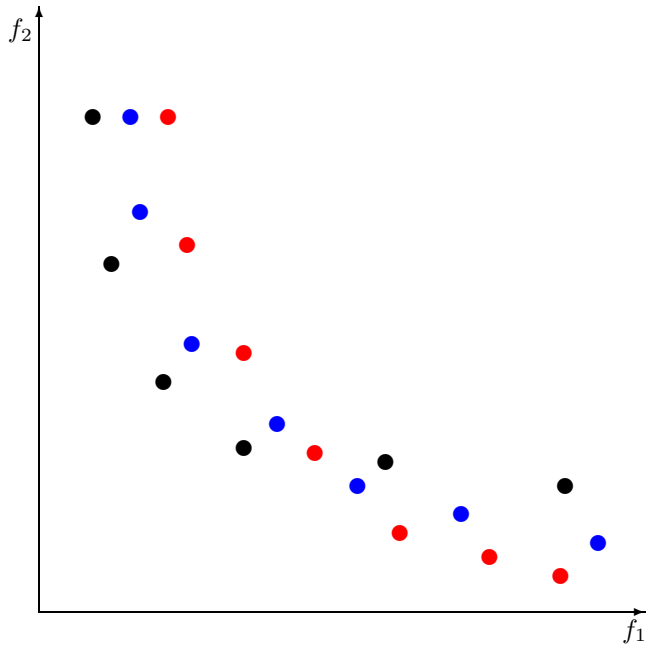

Figure 3.7: Three approximations are considered: the black approximation is the best for the first objective, the red approximation is the best for the second objective while the blue approximation is a good compromise between the black one and the blue one. These three approximations are incomparable.

Two approximations can be easily incomparable in term of Pareto dominance. For example, an approximation A composed of all the non-dominated points, except one, is incomparable to the approximation B that would be composed of that missing point. We cannot say that A is better (see Table 3.1) than B because the missing point is not weakly dominated by any of the non-dominated points of A. However, in practice, in the case when the preferences of the decision maker are not known, the approximation A would be preferred to the approximation B, because the chances that A meets the requirements of the decision maker are much higher than with B. Therefore, despite this theoretical incomparability between approximations, it is necessary to use tools to assess the quality of the approximations generated by MOMHs.

The representation in objective space of the approximations is a good way to evaluate the quality of approximations, but this technique is only efficient when the number of objectives is at most equal to three. Moreover, for MOCO problems, the number of potentially non-dominated points can be very high, and it could be difficult to evaluate the difference between two approximations on the basis of a representation in objective space.

Another problem is that MOMHs are generally stochastic and each run of a MOMH generates thus a different approximation. One way to deal with that is to use the attainment function approach [94], which consists in estimating the probability of attaining arbitrary goals in objective space from multiple approximation sets. To evaluate the quality of the results generated by different MOMHs, we compare this probability density function in objective space. This approach is neat, but the problem of comparing approximations based on a graphical comparison still exists, and it is unclear how to use quality measures in the attainment function framework [140].

The approach to compare approximations considered in this work is simple but well-approved in the MO community.

We will first use unary quality indicators. A unary quality indicator I is defined as a mapping from the set of all approximation sets ψ to the set of real numbers:

$$I : \psi \rightarrow \mathbb{R}.$$

It is difficult to define a unique unary quality indicator given that many aspects count, in particular the proximity of the approximation to the Pareto front, the distribution of the potentially non-dominated points, the repartition of these points, etc. Therefore, we will use several unary quality indicators.

To take into account the stochasticity of the MOMHs, we will compare the average of the values of the indicators over a certain number of runs. In addition, we will perform the Mann-Whitney test to take into account the variations in the results provided by the algorithms. Outperformance relations, that consist in comparing the points of the approximations in term of Pareto dominance, will also be realized.

We first present the different unary quality indicators used in this work (sections 3.5.1 and 3.5.2). For some of these indicators, reference sets will be necessary and we will introduce the notion of *ideal* set with this aim (section 3.5.3). The Mann-Whitney test (section 3.5.4) and the outperformance relations (section 3.5.5) are then briefly presented.

3.5.1 Indicators used in this work

Without loss of generality, we will present the indicators in the case of a minimization MOP. We want to measure the quality of the approximation $\widetilde{X}_E$ of the efficient set. We will use these notations and definitions:

- y^0: a reference point such that $y^0 \preceq f(x)\ \forall\, x \in \mathcal{X}$.
- y^1: a bounding point such that $y^1 \succeq f(x)\ \forall\, x \in \mathcal{X}$.
- Y_R: a reference set composed of points such that $\forall\, x \in \mathcal{X},\ \exists y \in Y_R \,|\, y \preceq f(x)$. In other words, Y_R weakly dominates any sets composed of solutions $x \in \mathcal{X}$.
- $\widetilde{Y}_N$: an approximation of the Pareto front, image in objective space of $\widetilde{X}_E$.

Hypervolume (to maximize)

The hypervolume $\mathcal{H}$ has been introduced by Zitzler in 1999 [249]. The hypervolume indicator measures the (hyper)volume of the portion of the objective space that is weakly dominated by an approximation $\widetilde{Y}_N$. The objective space must be bounded, otherwise a bounding point y^1 has to be used. The higher the value of the hypervolume, the better the approximation is.

The representation of the hypervolume for a biobjective minimization problem is illustrated in Figure 3.8.

The constraint about the bounding point, that is $y^1 \succeq f(x)\ \forall\, x \in \mathcal{X}$, is often too restrictive. Therefore, as bounding point, we can use the nadir point multiplied by a certain coefficient, determined experimentally. However, the bounding point must remain at least weakly dominated

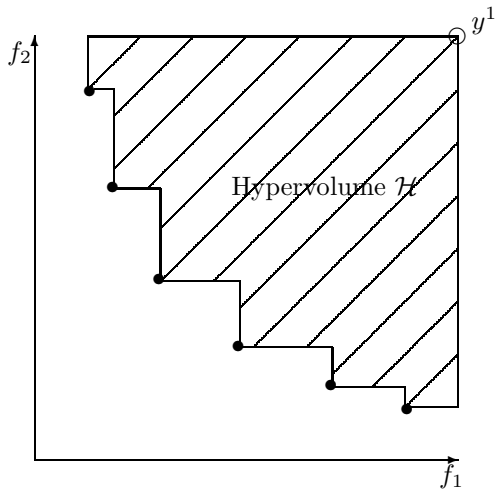

Figure 3.8: Illustration of the hypervolume indicator for a biobjective minimization problem.

by all the points of the approximations for which the hypervolume is computed. Otherwise, the value of the hypervolume would be skewed.

A drawback of the hypervolume is its computation which is exponential in the number of objectives [244]. But many authors [21, 27, 247] have recently developed efficient algorithms for computing the hypervolume, even when the number of objectives is high.

R indicator (to minimize)

The R indicator has been introduced by Hansen and Jaszkiewicz in 1998 [100] and is defined as the estimation of the expected value of weighted Tchebycheff scalarizing function on approximation $\widetilde{X}_E$ over the set Λ of normalized weight vectors:

$$R(\widetilde{X}_E, y^0, \Lambda) = \frac{1}{|\Lambda|} \sum_{\lambda \in \Lambda} \min_{x \in \widetilde{X}_E} ||f(x) - y^0||_\lambda \tag{3.5}$$

where y^0 is the reference point and Λ is a set of normalized weight sets ($\sum_{k=1}^{p} \lambda_k = 1, \lambda_k \geq 0$).

This measure has also been normalized by Jaszkiewicz [118] such that an approximation composed of the reference point y^0 obtains a R value equal to 1.

The values of the objectives of the approximation are first normalized in the following way:

$$f'_k(x) = \frac{y^1 - f_k(x)}{y^1 - y^0} \quad \forall\, k \in \{1, \ldots, p\},\ \forall\, x \in \widetilde{X}_E$$

The computation of R normalized, called R_N is then done as follows:

$$R_N(\widetilde{X}_E, y^0, y^1, \Lambda) = 1 - \frac{1}{|\Lambda|} \sum_{\lambda \in \Lambda} \min_{x \in \widetilde{X}_E} ||f'(x) - y'^0||_\lambda \tag{3.6}$$

where the point y'^0 is the point $(1, 1, \ldots, 1)$.

Binary ϵ-indicator I_ϵ (to minimize)

The binary ϵ-indicator I_ϵ has been introduced by Zitzler *et al.* [255] and is based on the ϵ-dominance relation. If we consider two approximations A and B, the ϵ-indicator I_ϵ gives the

ϵ factor by which the approximation A set is worse (if $\epsilon > 1$) or better (if $\epsilon < 1$) than the approximation B with respect to all objectives:

$$I_\epsilon(A, B) = \inf_{\epsilon \in \mathbb{R}^+} \{\forall\, b \in B, \exists\, a \in A : a \preceq_\epsilon b\} \tag{3.7}$$

This indicator can also be used as a unary indicator if we use as set B the reference set Y_R. We will denote this unary indicator by $I_{\epsilon 1}$. A value smaller than 1 would imply that A strictly dominates the reference set.

Distance D_1 and D_2 (to minimize)

The indicators D_1 and D_2 have been introduced by Czyzak and Jaszkiewicz [35] and Ulungu *et al.* [233]. They allow to measure the average distance and maximal distance between the reference set Y_R and the approximation $\widetilde{Y}_N$. Ideally, the reference set is the Pareto front Y_N.

If we consider any distance $d\big(y^1, y^2\big)$ between two points y^1 and y^2 in the objective space, we can define the distance $d'(\widetilde{Y}_N, y^1)$ between a point $y^1 \in Y_R$ and the points $y^2 \in \widetilde{Y}_N$ as follows:

$$d'(\widetilde{Y}_N, y^1) = \min_{y^2 \in \widetilde{Y}_N} d(y^2, y^1)$$

This distance is thus equal to the minimal distance between a point y^1 of Y_R and the points of the approximation $\widetilde{Y}_N$.

According to that, we can consider two points of view:

- The mean distance D_1 between $\widetilde{Y}_N$ and Y_R:

$$D_1(\widetilde{Y}_N, Y_R) = \frac{1}{|Y_R|} \sum_{y^1 \in Y_R} d'(\widetilde{Y}_N, y^1) \tag{3.8}$$

- The maximal distance D_2 between $\widetilde{Y}_N$ and Y_R:

$$D_2(\widetilde{Y}_N, Y_R) = \max_{y^1 \in Y_R} d'(\widetilde{Y}_N, y^1) \tag{3.9}$$

In this work, to evaluate the distance $d\big(y^1, y^2\big)$ between two points y^1 and y^2, we use the weighted Euclidean distance:

$$d(y^1, y^2) = \left(\sum_{k=1}^{p} w_k (y^1 - y^2)^2 \right)^{\frac{1}{2}}$$

where the w_k values are weights with the aim of taking into account the ranges of the objectives. They are computed as follows:

$$w_k = \frac{1}{y_k^N - y_k^I} \quad k = 1, \ldots, p$$

Proportion of non-dominated points (to maximize)

If the non-dominated points of the problem are known, we can compute the proportion of non-dominated points generated:

$$P_{Y_N} = \frac{|\widetilde{Y}_N \cap Y_N|}{|Y_N|} \tag{3.10}$$

And if the distinctions between supported and non-supported non-dominated points is realized, we can compute the proportion of supported non-dominated points:

$$P_{Y_{SN}} = \frac{|\widetilde{Y}_N \cap Y_{SN}|}{|Y_{SN}|} \tag{3.11}$$

and the proportion of non-supported non-dominated points:

$$P_{Y_{NN}} = \frac{|\widetilde{Y}_N \cap Y_{NN}|}{|Y_{NN}|} \tag{3.12}$$

To measure the proximity of the approximation $\widetilde{Y}_N$ to the Pareto front Y_N, we can also compute the proportion of points of $\widetilde{Y}_N$ located in the right triangles defined between two consecutive supported non-dominated points (for biobjective problems) [233]. If $Y_\triangle$ denotes the region of the objective space generated by the triangles (see Figure 3.9), the proportion $P_{Y_\triangle}$ of points in the triangles is defined as follows:

$$P_{Y_\triangle} = \frac{|\widetilde{Y}_N \cap Y_\triangle|}{|Y_N|} \tag{3.13}$$

As supported and non-supported non-dominated points of $\widetilde{Y}_N$ are located in $Y_\triangle$ (the borders of the triangles are taken into account), we have $P_{Y_\triangle} \geq P_{Y_{SN}}$ and $P_{Y_\triangle} \geq P_{Y_{NN}}$.

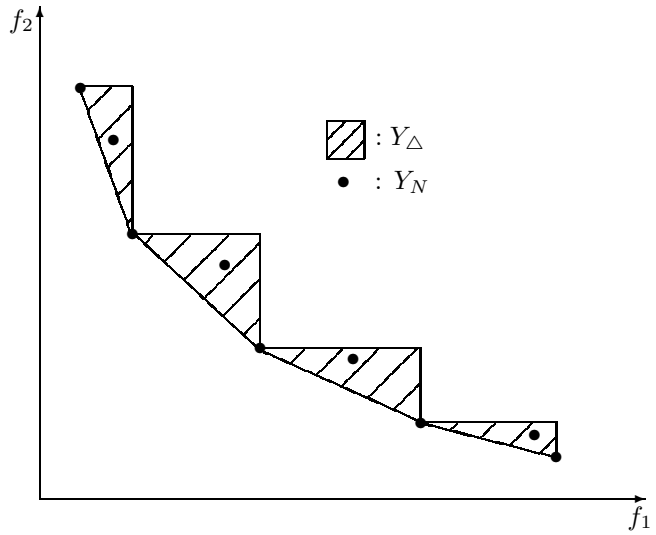

Figure 3.9: Illustration of the region of the objective space generated by the right triangles defined between two consecutive supported non-dominated points.

The number of potentially efficient solutions will also be indicated, but not used as an indicator, since this measure is not really significant. This number will be denoted by $|PE|$.

3.5.2 Powerfulness of the indicators considered

It would be great if with an indicator I or even with a finite combination $\mathbf{I}$ of unary indicators I, the following equivalence held:

$$\mathbf{I}(A) < \mathbf{I}(B) \Leftrightarrow A \lhd B$$

Such an equivalence would allow to easily conclude than an approximation A is better than an approximation B or to certify that an approximation A is better than an approximation B.

Unfortunately, this equivalence does not exist, as shown by Zitzler *et al.* [255].

We have however the following relations:

- $A \lhd B \Rightarrow \mathcal{H}(A) > \mathcal{H}(B)$.
- $A \prec\prec B \Rightarrow I_{\epsilon 1}(A) < I_{\epsilon 1}(B)$ and $A \lhd B \Rightarrow I_{\epsilon 1}(A) \leq I_{\epsilon 1}(B)$.

- $A \prec\prec B \Rightarrow R(A) < R(B)$ and $A \lhd B \Rightarrow R(A) \leq R(B)$.

However, none of these unary indicators allows to affirm that $A \lhd B$, that is we cannot inverse the preceding relations. For example, we can have two incomparable approximation A and B with $\mathcal{H}(A) < \mathcal{H}(B)$. A good example to illustrate that is given by Lizárraga *et al.* [153] in Figure 3.10. In this figure, five incomparable approximations are represented. If we compute the hypervolume $\mathcal{H}$ of each of these approximations, we have $\mathcal{H}(C) > \mathcal{H}(B) = \mathcal{H}(D) > \mathcal{H}(A) = \mathcal{H}(E)$ even if the five approximations are incomparable.

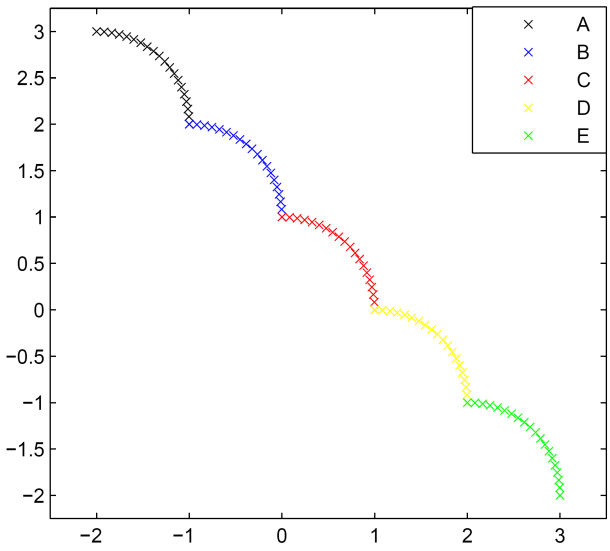

Figure 3.10: Illustration of five incomparable approximations. The approximation in red has a better hypervolume indicator that the other ones even if all approximations are incomparable [153].

We can only say (that follows from the preceding relations):

- $\mathcal{H}(A) < \mathcal{H}(B) \Rightarrow A \not\lhd B$.
- $I_{\epsilon 1}(A) > I_{\epsilon 1}(B) \Rightarrow A \not\lhd B$.
- $R(A) > R(B) \Rightarrow A \not\lhd B$.

That means that if the indicators $\mathcal{H}$, $I_{\epsilon 1}$ or R of an approximation A is worse than for an approximation B, we can infer that A cannot be better than B.

For the percentage indicators P_{Y_N}, $P_{Y_{SN}}$ and $P_{Y_{NN}}$, we have:

- $P_{Y_N}(A) < P_{Y_N}(B) \Rightarrow A \not\lhd B$.
- $P_{Y_{SN}}(A) < P_{Y_{SN}}(B) \Rightarrow A \not\lhd B$.
- $P_{Y_{NN}}(A) < P_{Y_{NN}}(B) \Rightarrow A \not\lhd B$.

These relations are justified by the fact that once non-dominated points belong to an approximation, we cannot find another approximation with less non-dominated points better than this approximation.

For the $P_{Y_\triangle}$ indicator, nothing can be say since the points in the triangles can be all dominated points.

It is also the case for the indicators D_1 and D_2. In Figure 3.11, we illustrate the problem with these indicators. The reference set is composed of the points $(2, 1)$ and $(1, 8)$, the approximation

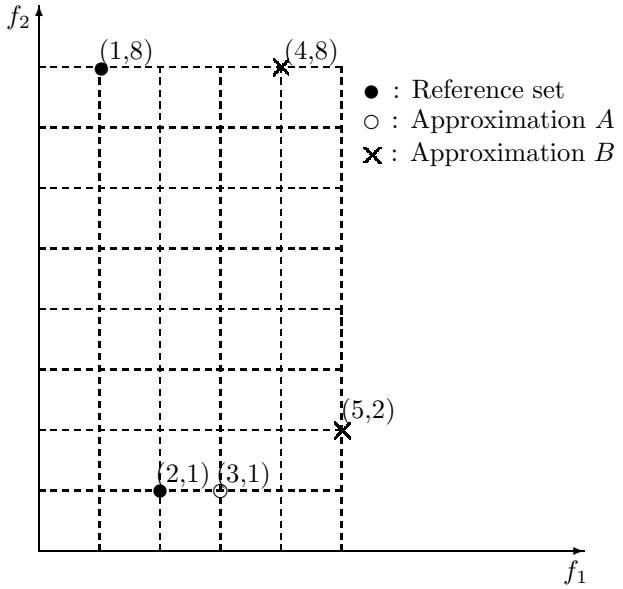

Figure 3.11: Illustration of the problem with the D_1 and D_2 indicators: $A \prec\prec B$ but $D_1(A) > D_1(B)$ and $D_2(A) > D_2(B)$.

A of the point $(3, 1)$ and the approximation B of the points $(5, 2)$ and $(4, 8)$. We have thus that A strictly dominates B.

If we compute the D_1 and D_2 distances of A and B, we obtain:

- $D_1(A) = 4.14$ and $D_1(B) = 3.08$.
- $D_2(A) = 7.28$ and $D_2(B) = 3.16$.

We see that $D_1(B) < D_1(A)$ and $D_2(B) < D_2(A)$ even if $A \prec\prec B$. However, in practice, the case where $A \prec\prec B$ with $D_1(B) < D_1(A)$ rarely occurs, since the sets are normally composed of a well-distributed set of points. The distance D_2 is more sensible to this problem since D_2 only depends on one point (the one that has the maximal distance to the reference set).

In conclusion, with the unary $\mathcal{H}$, $I_{\epsilon 1}$, R, P_{Y_N}, $P_{Y_{SN}}$ and $P_{Y_{NN}}$ indicators, we will not be able to say that an approximation A is better than an approximation B but just A is not worse than B, that means that either A is better than B or incomparable to B.

Even though, an approximation A that finds better values for the indicators used in this work is often preferred to an approximation B, since we consider that these indicators reflect well the preferences of the decision maker.

3.5.3 Reference set

As said before, to compute the D_1, D_2 and $I_{\epsilon 1}$ indicators, it is important to have a reference set Y_R with the following property:

$$\forall\, x \in \mathcal{X},\ \exists y \in Y_R \,|\, y \preceq f(x)$$

Two typical techniques to compute this set are as follows [140]:

- Generation of the Pareto front.
- Merging of all the approximations generated by state-of-the-art algorithms. The dominated points are removed from this union and the remaining points, which are not dominated by any of the approximations, form the reference set.

To have the Pareto front as reference set would be perfect, but is often very time-consuming and impossible to generate in most of the cases.

One advantage of the second approach is that the reference set weakly dominates all the approximations under consideration. But that could not be the case anymore if a new approximation is compared to this reference set. In this case, the reference set should be normally re-computed whenever additional approximations are included in the comparison, which is not very convenient. Furthermore, good state-of-the-art results have to be published, which is not the case for all MOPs.

As both techniques are not satisfactory, we introduce the notion of *ideal set* for biobjective problems with $\mathcal{Y} \subset \mathbb{Z}^2$, that is the values of the objectives are integer. The ideal set is a lower bound of the Pareto front [57] and is defined as the best potential Pareto front that we can produce from a set of extreme supported non-dominated points. Extreme supported non-dominated points are used since these points are easier to generate than non-extreme supported non-dominated points and non-supported non-dominated points (no enumeration algorithm is needed, see section 3.2.3). For instance, we can see in Figure 3.12 the representation of five extreme supported non-dominated points of a biobjective problem (filled black points). This set can only be improved by adding non-extreme supported non-dominated points or non-supported non-dominated points. As $\mathcal{Y} \subset \mathbb{Z}^2$, it is easy to compute the best places than non-dominated points can possibly take. The coordinates of ideal non-extreme supported non-dominated points are the integer values located on the line between two consecutive extreme supported non-dominated points, and the coordinates of ideal non-supported non-dominated points are the integer values located the closest possible to this line. In Figure 3.12, we have added these ideal non-extreme supported non-dominated points and ideal non-supported non-dominated points, represented by the circles.

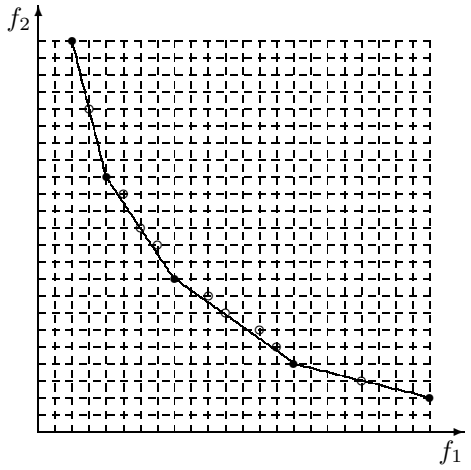

Figure 3.12: Ideal set produced on the basis of five extreme supported non-dominated points.

Hence, it is impossible to improve this set with feasible solutions, and that is why this set is called ideal set. It gives an excellent lower bound of the Pareto front. At final, to goes from one solution to another, only a step of one unit is produced, for the first or second objective, depending on the gradient of the line between two consecutive extreme supported non-dominated points.

All feasible solutions are weakly dominated by a solution of the ideal set. In addition, the ideal set has the advantage of being fixed, and does not have to be re-computed each time a new approximation is generated. It is thus an interesting property and the distance between the ideal set and an approximation will not be biased.

An obvious disadvantage of the ideal set is that the cardinality of the set can be huge. A crude upper bound of this cardinality is equal to $\max(y_1^N - y_1^I + 1, y_2^N - y_2^I + 1)$. If we denote by x^1 and x^2 two lexicographic optimal solutions of the biobjective problem, respectively equal to

$\text{lexmin}_{x\in\mathcal{X}}\big(f_1(x), f_2(x)\big)$ and $\text{lexmin}_{x\in\mathcal{X}}\big(f_2(x), f_1(x)\big)$), to attain this upper bound, one property must be met: $f_1(x^1) + f_2(x^1) = f_1(x^2) + f_2(x^2)$ such that the line drawn between $f(x^1)$ and $f(x^2)$ has a linear coefficient equal to -1. As a result, $f(x^1)$ and $f(x^2)$ are the only two extreme non-dominated points of the biobjective problem.

Nevertheless, it is not necessary to save all the solutions of the ideal set, as the ideal set is easy to compute. Also, the distance between the solutions of the ideal set and the solutions of an approximation can be exactly generated quickly as it is easy to sort the solutions of a non-dominated set according to one of the objectives by using a simple propriety of a non-dominated set in the biobjective case: for two solutions x^1 and x^2 belonging to a non-dominated set, $f_1(x^1) < f_1(x^2)$ implies that $f_2(x^1) > f_2(x^2)$. Therefore, no computational difficulty due to the high cardinality of the ideal set has appeared during the experimentations realized in this work.

3.5.4 Mann-Whitney test

The Mann-Whitney test will be applied in order to statistically compare the values obtained by an indicator for the different runs of two MOMHs.

The Mann-Whitney test [69, 166] is a non-parametric statistical test assessing whether two independent samples of observations come from the same distribution. This test uses the ranks of the values taken by the observations put together.

We use a non-parametric statistical test since we do not know the distributions of the indicators.

We test the following hypothesis: "the two samples come from identical populations" for a specific indicator on a given instance. When the hypothesis is satisfied, the sign "=" will be indicated (no difference between the values of the indicators obtained by the algorithms). When the hypothesis is not satisfied, there are differences between the indicators of the algorithms and normally the sign "≠" should be indicated. However, we will be more precise and the sign ">" will be indicated if the mean value obtained with an algorithm is better than the mean value obtained with another algorithm and the sign "<" will be indicated otherwise.

As many hypothesis are tested simultaneously (equal to the number of indicators for which the Mann-Whitney test is performed), the levels of risk α of the tests have been adjusted with the Holm sequential rejective method [105]. This method works as follows: the p-values obtained with the Mann-Whitney test for each indicator are sorted by increasing value and the smallest p-value is compared to α/k (where k is the number of indicators). If that p-value is less than α/k, the hypothesis related to this p-value is rejected. The second smallest p-value is then compared to $\alpha/(k-1)$, the third to $\alpha/(k-2)$, etc. The procedure proceeds until each hypothesis has been rejected or when a hypothesis cannot be rejected.

3.5.5 Outperformance relations

With outperformance relations [100], a pairwise comparison is realized between the points of two approximations A and B.

Four cases can occur: a point of A is dominated by at least one point of B, a point of A dominates at least one point of B, a point of A is equal to another point of B, or the result of the comparison belongs to none of these three possibilities (the point of A is incomparable to all the points of B). For each of the four cases, the percentage of points of the approximation A that check the case is computed.

The outperformance relations will be represented with box-plot graphs, that allow to represent the variations in the different values (as a pairwise comparison is made between the runs of two distinct algorithms).

3.6 Summary

In this relatively long chapter, we have looked over the different methods for solving MOPs. We have started with the exact methods where we have laid the stress on the two-phase method, for mainly two reasons: this method has been adapted to a high number of MOCO problems and, in chapter 6, we will present a heuristic adaptation of the first phase of this method.

Approximation algorithms have been then presented, but only briefly since few works have been undertaken in this stream of research.

The main section of this chapter has been after developed: metaheuristics for MOO. The aim of this section was not to expose all the existing adaptations of metaheuristics, but to present the different elements common to MOMHs through four models: MOLS1, MOLS2, MOGA and MOMA.

We are entirely aware that with this review, many adaptations of metaheuristics to MO have been left out (adaptations based on ACO, GRASP, VNS, etc.) and that all MOMHs do not follow one of the four models considered.

The base for the understanding of the next chapters has however been presented.

In this last section of this chapter, we have shown an important point in the framework of this work, that is on what the assessment of the performances of the MOMHs studied in this work will be based.

4

MOKP: state-of-the-art

We present in this chapter a state-of-the-art of one classical MOCO problem studied in this work: the multiobjective version of the knapsack problem (KP). This problem has been widely studied, and the multidimensional case (MOMKP) as much as the unidimensional case (MOKP) (see section 2.2.4 in chapter 2 for a presentation of these problems). For the MOKP, exact methods have been first developed. On the other hand, as far as we know, no exact method has been adapted to solve the MOMKP. The MOMKP is often used to test new MOMHs: several experimental comparisons of different approaches to solve the MOMKP have been reported in the literature. One of the reasons of this success is that Zitzler [253] in 1998 used the MOMKP to test his new MOMH, and published the instances and the results obtained. From that moment, many researchers developed new MOMHs and tried to obtain better results: the race was open!

In this review of the literature, we first relate the works about the MOKP since this problem is the basis of the MOMKP. We present then the works concerning the MOMKP.

4.1 The MOKP

We decompose the presentation of the methods in three groups: exact methods, approximation algorithms and heuristic methods based on metaheuristics. We end this section by reviewing particular studies of this problem.

4.1.1 Exact methods

Exact methods are known to solve efficiently this problem. We cite and present briefly these methods and we give the size of the instances (that is the number of items) that the methods can deal with.

- The two-phase method (see section 3.2.3 in chapter 3) was adapted to this problem by Visée *et al.* [240] in 1998. In the second phase of the procedure, they used a branch and bound algorithm with the aim of generating the non-supported efficient solutions. They considered randomly generated biobjective instances. The value of W was equal to half the value of the total weight ($\sum_{i=1}^{n} w_1^i$); that will be the case for all instances of the MOKP presented in this section. They solved instances with up to 500 items.
- In 2000, Klamroth and Wiecek [135] presented several dynamic programming formulations, but unfortunately no computational result was reported.
- Captivo *et al.* [29] proposed in 2003 an exact method based on a transformation of the MOKP into a biobjective shortest path problem. The biobjective shortest path problem was solved with an adapted version of the labeling algorithm of Martins *et al.* [169]. They used biobjective instances of three different types: random, weakly correlated and strongly correlated. They solved the random instances with up to 320 items, the weakly correlated instances with up to 800 items and the strongly correlated instances with up to 900 items. They showed that they obtain better results in term of computational time than the approach of Visée *et al.*

- In 2006, Figuera *et al.* [70] extended the results of Captivo *et al.* and of Klamroth and Wiecek and presented algorithms for constructing four network models, all representing the MOKP and transforming the MOKP into the multiobjective shortest path problem. They tested the formulations on different types of instances but the results were not reported.
- In 2007, Jolai *et al.* [125] presented a new exact method based on a linear programming formulation but no computational result was presented.
- Jorge and Gandibleux [127] proposed in 2007 an improvement of the two-phase method of Visée *et al.*. They ameliorated the second phase by improving the bounds of the branch and bound. They also proposed the use of a ranking algorithm in the second phase. They showed that they improve the results of *Visée et al.*
- Recently, in 2009, Bazgan *et al.* [18] presented an exact method based on dynamic programming. By using several complementary dominance relations to discard partial solutions, they developed a very efficient method. They tested their method on different types of instances: random (type A), strongly correlated (type B), weakly uncorrelated (type C) and strongly uncorrelated (type D) biobjective instances and on three-objective instances of type A and C. In less than two hours of computationnal time on a bi-Xeon 3.4 GHz CPUs with 3072 MB of RAM, they solved biobjective instances of type A with 700 items, of type B with 4000 items, of type C with 500 items and on type D with 250 items. They compared their results with the method of Captivo *et al.* and with an ϵ-constraint method coupled with the ILOG Cplex 9.0 solver, and they obtained better results and solved instances of higher size. For the three-objective instances, due to the explosion of the cardinality of X_E they could only solve instances of type A with up to 110 items and instances of type C with up to 60 items, which is still remarkable since this is the first exact method that was adapted to three-objective instances of the MOKP.

4.1.2 Approximation algorithms

We present here approximation methods aiming to produce PTAS and FPTAS (see section 3.3 in chapter 3).

- Erlebach *et al.* [67] presented in 2002 an efficient and applicable FPTAS for the MOKP. They also presented a PTAS for the MOMKP based on linear programming.
- In 2006, Kumar and Banerjee [143] presented a stochastic algorithm to obtain $(1+\epsilon)$-approximations. They called their algorithm REMO for restricted evolutionary multiobjective optimizer. REMO is based on a restricted mating pool with a separate archive to store the remaining population. They presented a rigorous running time analysis of the algorithm.
- Recently, in 2009, Bazgan *et al.* [17] proposed a new FPTAS with a practical behavior. As in their exact method [18], the main idea of their approach is based on dynamic programming and relies on the use of several complementary dominance relations to discard partial solutions. They tested their approximation algorithm on the same four type instances (A, B, C and D) that they used to test their exact method (see the previous subsection). With this approximation algorithm and with an ϵ value equal to 0.1, they solved biobjective instances A with up to 2500 items, instances B with up to 20000 items, instances C with up to 2000 items and instances D with up to 900 items. Regarding the three-objective instances, they solved, with an ϵ value equal to 0.1, instances A with up to 250 items and instances C with up to 140 items. They compared their FPTAS with the one of Erlebach *et al.* and they showed that they obtain better results.

4.1.3 Heuristic methods

We present now a review of the heuristic methods with the aim of finding a good approximation of the efficient set.

- In 1993, Ulungu [228] presented in his thesis the first adaptation of SA to MO through the MOSA method (see section 3.4.1 in chapter 3). He adapted the method to tackle the MOKP (see also [233]). He solved random biobjective instances with up to 500 items (the same instances than in the exact method of Visée *et al.*).

- Ben Abdelaziz and Krichen [2] proposed in 1997 a tabu search-based method. They solved random biobjective instances with up to 100 items. They extended their method in 1999 by integrating the tabu search into a genetic algorithm [1].

- In 1998, Czyzak and Jaszkiewicz [35] proposed the Pareto simulated annealing (PSA) (see section 3.4.1 in chapter 3) to solve the MOKP. They solved random biobjective, three-objective and four-objective instances with up to 800 items.

- Gandibleux and Freville [75] proposed in 2000 a tabu search method, called MOTS (see section 3.4.1 in chapter 3). They used different techniques to reduce the decision space to an interesting area. They tested their method on random biobjective instance of 50 and 100 items. No comparison with other heuristic methods has been realized.

- Gandibleux *et al.* [78] in 2001 developed a two-phase method where in the first phase an exact algorithm to compute the supported efficient solutions has been used and in the second phase the traditional components of a genetic algorithm to improve the initial population has been applied. They used the same instances than Visée *et al.* but no comparison with other methods has been undertaken.

- Zhang and Ong [248] proposed in 2004 a simple method essentially based on an efficient heuristic for the linear relaxation. They tested their method on random biobjective instances. They solved instances from 500 to 50000 items and they showed that they obtain better results than if their method is coupled with the ILOG CPLEX 7.5 solver.

- Gomes da Silva *et al.* studied this problem extensively. They first proposed in 2006 [37] a scatter search (SS) [87] based method. They followed the usual structure of SS, composed of five steps:

  1. Diversification (which creates a collection of trial solutions).
  2. Improvement (which transforms the trial solutions into enhanced ones, and usually restores feasibility).
  3. Reference set update method (which maintains the reference set with the best solutions according to certain criteria).
  4. Subset generation (which creates subsets of solutions from the reference set).
  5. Solution combination method (which combines solutions from each subset to create new ones).

  They tested their method on large size random biobjective instances with a number of items going from 100 to 6000. They compared their method to the exact method of Visée *et al.* and showed that for the 100 items instance, they generate in average 33.13% of the efficent solutions, for the 300 items instance, 9.75% and for the 500 items instance, 5.12%. But obvioulsy, the running time of their method is much lower than the exact method and when $n = 500$, the exact method takes about 400 times the computational time required by theur heuristic SS method.

  In 2007 [39], they presented an improvement of their previous SS method, by modifying the combination method which induces changes in the reference set update method and the

subset generation method. In the combination method, they used an adapted version of the exact method of Visée *et al.* to solve the residual problems that are small size problems. They tested their method on the same instances than in their previous work. They also compared the results obtained to the efficient set obtained with the exact method of Visée *et al.* They improved their previous results since they generate in average 87.3% of the efficient solutions for the 100 items instance, 95.0% for the 300 items instance and 91.2% for the 500 items instance. On the other hand, the improvement of running time was not anymore spectacular and for the 500 items instance, the running time of the exact method takes only 1.5 times the computational time required by the new SS method. They also compared their results with other metaheuristics: the MOSA method of Ulungu *et al.* and the two-phase method of Gandibleux *et al.* presented above. They showed that they obtain better results.

4.1.4 Particular studies

We finish the review of the works concerning the MOKP by four interesting particular studies.

- In 2006, Gandibleux and Klamroth [76] studied cardinality bounds for the MOKP based on weighted sums scalarizations. They showed that we can use these bounds to reduce the feasible set of the biobjective MOKP.
- In 2007, Gandibleux and Ehrgott [57] introduced the concept of bound sets for MOCO problems. Indeed, well-known bounds for multiobjective problems are the ideal point (lower bound) and the nadir point (upper bound) (see section 2.1.5 in chapter 2) but first of all, these bounds are not easy to compute, especially the nadir point, and secondly, these values are very far from the Pareto front. They thus generalized the notion of a bound value to that of a bound set and applied these concepts to, among others, the MOKP. They obtain upper bound set by using the linear relaxation of the MOKP and lower bound set by using a simple greedy algorithm.
- Gomes da Silva *et al.* recently studied, in 2008, the interesting notion of core problems (see section 2.2.4 in chapter 2) for the MOKP [38]. Indeed, the most efficient algorithms for solving the single-objective KP are based on the core concept but this concept has never been directly used in the MOKP. They thus investigated the existence of the core structure in the MOKP and defined the notion of *biobjective* core. They reported computational experiments related to the size of the biobjective core on different types of instances. The results show that, on average, the biobjective core is a very small percentage of the total number of items. Then, they proposed a heuristic and an exact method based on these results, but without experimenting these methods.
- In 2007, Jorge *et al.* [127] presented new properties aiming to a priori reduce the size of the biobjective MOKP. Based on a lower and upper bound on the cardinality of a feasible solution for KP introduced by Glover in 1965 [83] and on dominance relations in the space of data of the MOKP, they reduced the size of the biobjective instances of the MOKP by fixing, a priori, about 10% of the variables, on random instances.

4.1.5 Summary

To summarize, the exact method and approximation method of Bazgan *et al.* [17, 18] seem very competitive and an accurate comparative approach between the methods of Bazgan *et al.* and the methods of Gomes da Silva *et al.* [37, 39] would be relevant.

4.2 The MOMKP

For the MOMKP, as far as we know, only heuristic methods have been developed. As the methods are most of the time hybrid methods, we do not make a distinction between the

method based on EAs and the methods based on LS. We prefer to give a chronological review. After presenting these heuristic methods, we review particular studies of this problem.

4.2.1 Heuristic methods

- It seems that is first Zitzler and Thiele [254] who tackled the multidimensional version of the MOKP, in 1999. Zitzler and Thiele performed a comparative study of five different MOGAs for the MOMKP, and also introduced SPEA (see section 3.4.2, chapter 3). They showed that this method outperforms the other methods. In this paper, they introduced the instances of the MOMKP that will be used by many other authors later. For these instances (that we will call the ZMKP instances), the number of objectives is equal to the number of constraints. Nine different instances with two, three and four objectives, in combination with 250, 500 and 750 items have been created. The profits and the weights were randomly generated in the interval [10,100]. The knapsack capacities were set to half the total corresponding weight. In all the works presented below, if nothing is mentioned for the instances used, that means that the ZMKP instances are considered.

- In 2000, Knowles and Corne [137] compared M-PAES [138] (see section 3.4.3 in chapter 3), based on Pareto ranking of the solutions with RD-MOGLS [112], based on random scalarization functions. They showed that both algorithms work well and produce better results than the (1+1)-PAES method [136].

- In 2000, Jaszkiewicz [114] applied the MOGLS method (see section 3.4.3 in chapter 3) and compared it to SPEA. He showed that MOGLS outperforms SPEA. In 2001, Jaszkiewicz continued his experiments and in [116], he compared five algorithms: MOGLS, M-PAES [137], SMOSA [209], MOSA [233] and PSA [35] (see section 3.4.1, chapter 3). Jaszkiewicz concluded that MOGLS outperforms the other methods. In [119], a comparison has been realized between MOGLS, SPEA, M-PAES and IMMOGLS [109] (see section 3.4.3 in chapter 3) and he also concluded that MOGLS outperforms the other methods. Jaszkiewicz published the results obtained with the MOGLS method and these results became the new reference for testing new methods.

- An improved version of the SPEA algorithm, called SPEA2 (see section 3.4.2, chapter 3), was further developed by Zitzler *et al.* in 2001 [251]. The new algorithm has been tested on a subset of the ZMKP instances and the authors concluded that SPEA2 performs better than its predecessor on all instances. No comparison of this new method with MOGLS has been undertaken.

- In 2002, Barichard and Hao [15] proposed a tabu search method (see section 3.4.1, chapter 3) that gives better results than SPEA but worse results than MOGLS. They improved in 2003 the tabu search by integrating it in a genetic algorithm [14] and with this new hybrid method, they concluded that they obtain better results than the MOGLS method. Unfortunately, they have not published their results.

- In 2004, Jaszkiewicz [120] gave up his MOGLS algorithm and compared three MOMHs: his PMA method [115], SPEA and the controlled elitist non-dominated sorting genetic Algorithm (CENSGA) [44]. The PMA method has been adapted to the MOMKP in a more sophisticated way than the MOGLS method. The SPEA and CENSGA methods have also been adapted to the MOMKP by Jaszkiewicz in the same way as PMA. The three methods share thus some identical components. Jaszkiewicz showed that PMA performs better than SPEA and CENSGA on instances with more than two objectives. After this work, it seems that this adaptation of PMA became the new reference method, since this method is an evolution of MOGLS and is adapted with more sophisticated operators. Unfortunately, the results of PMA were not published, and the previous results obtained with MOGLS remained the reference, even if it was possible to generate the results of PMA with the source code of PMA that Jaszkiewicz published. Indeed, the

MOMHLib++ library, developed by Jaszkiewicz [113] has been published. This library makes it possible to run various existing MOMHs on the MOMKP. In this library, the MOGLS method has also been adapted following PMA and gives better results than the initial MOGLS method [114]. In the chapter 7 about the resolution of the MOMKP, this version of the MOGLS algorithm will be called MOGLS04.

- Li *et al.* [149] studied in 2004 a hybrid adaptation of the estimation of distribution algorithm (EDA) [146, 191] to the MOMKP, by using a local search based on weighted sums, a random repair method and a population sampled from a probability distribution. They compared their results with those of MOGLS and they showed that they obtain better results for some indicators.

- In 2004, Vianna and Arroyo [238] proposed an adaptation of the GRASP metaheuristic to the MOMKP. This metaheuristic, called GRASP for greedy randomized adaptive search procedure was introduced by Feo and Resende in 1989 [68], and developed after (see chapter 6 of [222] and chapter 8 of [89]). There are generally two phases within each iteration of the GRASP metaheuristic: in the first phase, an initial solution is constructed via an adaptive randomized greedy function and in the second phase a local search method is applied to the initial solution. The adaptation of GRASP to the MOMKP made by Vianna and Arroyo follows this frame: at each iteration a weighted linear sum of the objectives is defined and a solution is built considering this linear sum. The solution is then improved by a local search that also makes use of the linear sum.

  They compared their method to SPEA2 and MOGLS and showed that they obtain better results.

- Gomes da Silva *et al.* did a lot of work concerning the MOKP but they also tackled the MOMKP. Effectively, in 2004, Gomes da Silva *et al.* [36] presented the adaptation of the SS method, with surrogate relaxation, to the MOMKP. The method follows exactly the same frame than the SS method adapted to the MOKP presented in the previous section. However, surrogate relaxation [51, 84] was used to convert the MOMKP into a MOKP by generating adequate surrogate multipliers. A modified version of the SS method presented in [37] is then applied to the surrogate problem.

  They first compared their method with SPEA and MOGLS on the ZMKP instances with only two objectives. They showed by comparing in objective space the potentially non-dominated points generated that they obtain better results on these instances. They then tested their method on new instances, of bigger size: the number of items was included between 1000 and 3000 and the number of constraints went from 10 to 100. The values of the profits and the weights were randomly generated between 1 and 100 and each knapsack constraint capacity was equal to 50% of the sum of their weights. They evaluated the quality of their results by measuring different distances between the approximations obtained and the upper bound obtained with the surrogate relaxation. They showed that the approximations generated are very close to the upper bound. On the other hand, the running time on a Pentium 4 processor and 256 MB of RAM (the CPUs is not given), was close to one hour, but given the size and the number of constraints taken into account, that remains reasonable.

- Alves and Almeida [3] presented in 2007 a genetic algorithm based on the Tchebycheff scalarizing function, called multiobjective Tchebycheff genetic algorithm (MOTGA). Several stages were performed; each one intended for generating potentially non-dominated points in different parts of the Pareto front. Different weight vectors act as pivots to define the weighted Tchebycheff scalarizing functions and direct the search for each stage. As usual, two populations have been used: an internal population that participates in the genetic process and an external population which stores all the potentially non-dominated solutions found through the algorithm. Linear relaxation of the integer program based on weighted Tchebycheff scalarizing functions followed by a repair procedure has been used

to produce initial solutions. Then classic components from genetic algorithms has been employed to improve the initial population: tournament selection, crossover and mutation. The individuals were assessed by a fitness evaluation function based on the weighted Tchebycheff scalarizing function.

They compared MOTGA to SPEA, SPEA2 and MOGLS and they showed that MOTGA obtains better quality indicators than the other MOMHs, with a lower running time, even if the number of potentially non-dominated solutions obtained by MOTGA is not very high.

We end the review of the works concerning the MOMKP by two particular studies.

4.2.2 Particular studies of the MOMKP

- In 2007, Sato *et al.* [204] studied and compared the effects on performance of local dominance and local recombination applied with different locality. They used the NSGA-II algorithm [46] to run their experiments. They used the MOMKP to compare the different approaches and they showed that the optimum locality of dominance is different from the optimum locality of recombination. They also pointed out that the performances when local dominance and local recombination with different locality are applied, are significantly better than the performances when local dominance or local recombination is applied alone, or even when local recombination and dominance with the same locality are applied.

- Recently, in 2008, Beausoleil *et al.* [19] applied a multi-start search method combined with path-relinking to generate potentially efficient solutions in a *balanced* zone of the Pareto front. However, what they stand for balanced zone of the Pareto front is not defined at all and this term only appears in the abstract and in the conclusion! Moreover, they did not justify this approach. It seems that it means that the method focuses on solutions that establish a good compromise between all the objectives. The method is relatively complex and integrates many ideas developed by Glover [85, 88]: ghost image process, strategic oscillation approach and use of conditional-exclusion memory or developed by Glover and Laguna [90] with the path-relinking method. In addition, the method presents many parameters. The results of the method are compared to many MOGAs, including SPEA, SPEA2, NSGA and NSGA-II but not with MOGLS or PMA which gives better performances on the MOMKP. They showed that their approach is competitive regarding those algorithms.

4.2.3 Summary

To summarize, it is surprising and in some sense quite a pity that it is always the results of MOGLS that have been taken into consideration to evaluate the quality of new methods, since these results date from 2000. The reason is that the results of MOGLS have been published, but not the results of PMA with the most sophisticated operators. However, the software used by Jaszkiewicz to generate the results of PMA has been published and it was thus possible to use it to produce the results of PMA. That also allows to compare the computational time between different methods in a more straightforward way.

Moreover, except in the work of Gomes da Silva *et al.*, it is always the ZMKP instances that have been used. These instances being randomly generated, they do not represent well all possibilities of types of instances of the MOMKP (correlated, uncorrelated, etc.).

4.3 Particular variants of the MOKP

Kumar *et al.* [142] considered in 2008 a special case of the biobjective knapsack: one of the objectives is to maximize the total profit while the other is to minimize the total weight of the

selected items. They proposed for this problem a hyper-heuristic, that is in their case, a genetic programming system that evolves a set of heuristics and decides whether or not to add an item to the knapsack. They analyzed in detail the behavior of their algorithm.

5
MOTSP: state-of-the-art

We present in this chapter a state-of-the-art of the multiobjective TSP (MOTSP)(see section 2.2.5 in chapter 2 for a presentation of this problem).

One of the first papers concerning the MOTSP is the study of Borges and Hansen [23]. It is devoted to the global convexity (see section 2.2.3 in chapter 2) in MOCO problems in general, and the MOTSP in particular. The authors analyzed the landscape of local optima, considering the classical two-edge exchange neighborhood (two edges are deleted and replaced with the only possible pair of new edges that does not break the tour) and using well-known scalarizing functions like the weighted sum or the Tchebycheff function. They indicated the implications of the distribution of local optima in building MOMHs to approximate Y_N.

Almost all the existing methods are MOMHs.

In order to present the MOMHs suited to solve the MOTSP, we made a distinction between MOMHs based on EAs, MOMHs based on LS and MOMHs based on ant colony optimization (ACO). At the end of this chapter, we present particular studies and variants of the MOTSP.

The majority of the instances treated in the literature concerns the symmetric ($c_k(i,j) = c_k(j,i)$) biobjective case ($p = 2$); only sometimes p is equal to 3.

5.1 Evolutionary algorithms

We first present MOMHs essentially based on EAs.

- In 2002, Jaszkiewicz [118] applied MOGLS (see section 3.4.3 in chapter 3) to the MOTSP. To treat the MOTSP, he used a LS based on the standard two-edge exchange neighborhood. The recombination operator is the distance preserving crossover (DPX): all edges common to both parents are put in the offspring which is then completed by randomly selected edges to form a Hamiltonian cycle. Jaszkiewicz compared the performances to the results obtained by different MOMHs coming from his own implementation. The considered instances—with two or three objectives—are euclidean instances based on the TSPLIB library [198] and contain at most 200 cities. These instances will be used by many other authors. In these instances, the costs between the edges correspond to the Euclidean distance between two points in a plane. The locations of the points are randomly generated. We will call these instances the "Kro" instances. On these instances, but only those containing no more than 100 cities, Jaszkiewicz showed that MOGLS is very competitive and therefore the results of MOGLS will be later often used as reference by other authors, for comparison reasons.

- A MOGA is proposed by Yan *et al.* [246] in 2003.

  The main characteristics are that:

  – A reproduction operator, called "inver-over", is defined to replace the traditional crossover and mutation operators: an offspring is created from a parent by making use of a particular comparison with a solution randomly selected in the population.

  - Niches are defined with a sharing function (see section 3.4.2 in chapter 3) to preserve the diversity of the population.
  - Rather than assigning a fitness value to solutions, a boolean function "better" is defined to sort the individuals of the population, from the subset of best solutions to the subset of worst solutions.
  - To generate a new population, a competition is randomly chosen between two types: a "family competition", to replace the parent by the offspring if the second is better than the first; a "population competition" which compares the offspring with a randomly chosen solution in the population.

  Various two objectives instances are randomly generated and the results obtained with this new MOGA outperforms those obtained by a classical MOEA, SPEA (see section 3.4.2 of chapter 3).

- Kumar and Singh [144] proposed in 2007 a Pareto converging genetic algorithm (PCGA) which is a hybridization of a Pareto ranking GA with LS.

  Each solution of the population is evaluated by its rank equal to one plus the number of individuals dominating it such that all non-dominated individuals have a rank equal to one. Then two individuals are determined by a conventional roulette wheel selection and are crossed-over using the DPX operator to produce an offspring. The offspring is inserted into the population according to its rank against the whole set of individuals. Mutation is also applied. An original convergence criterion based on "rank-histograms" is defined. Two successive populations P_{t-1} and P_t are considered and the union $P_{t-1} \cup P_t$ is ranked. Taking each rank in turn, the histogram is the fraction of the solutions from P_t among those from $P_{t-1} \cup P_t$ having this rank. The perfect convergence corresponds to a rank histogram equal to a single non-zero entry of 0.5 for rank equal to 1, indicating that no solution superior to those in P_{t-1} has been generated in evolving the later generation P_t.

  The hybridization of PCGA is made with a LS based on the three-edge exchange neighborhood. To form the initial population, the LS is applied, with each objective taken separately, from individuals randomly generated. For the other populations, the LS considers the objectives simultaneously and uses Pareto ranking.

  Comparisons are made with the results of MOGLS and PD-TPLS [187] (see next section) on the biobjective Kro instances of 100 cities and, roughly speaking, the results appear equivalent between the three methods. However, no running time comparison is realized.

- The memetic random key genetic algorithm described in [203] by Samanlioglu *et al.* in 2008 is inspired by MOGLS. However, the crossover operator is different: a random number is associated to each city of each parent and the chromosome is represented alternatively by these numbers. A classical one-point crossover is applied but the random numbers are used as "sort keys" to decode the new solutions: the cities are sorted in the ascending order of their corresponding keys to indicate the travel order of the salesman. Unfortunately, the results are not compared with those of MOGLS, but only with the old results of Hansen [99], for which they however obtain better results. The instances used are the Kro instances of 100 cities.

- Elaoud *et al.* [63] proposed in 2008 a multiple crossover GA for the MOTSP. As many types of crossover and mutation operators have been proposed for the single-objective TSP (see [145]), all of them are included in the algorithm.

  At each iteration of the GA, a pair (crossover, mutation) is randomly selected based on a probability distribution. At the beginning, this distribution is uniform. Then, depending on the performance of the iteration—measured by the numbers of new solutions entering in $\widetilde{X}_E$ and solutions rejected from $\widetilde{X}_E$— the probability distribution is modified into a dynamic scheme to reward or not the selected pair of operators. Experimental results and comparison with those of MOGLS show the synergy effects among the different operators and prove the efficiency of the proposed approach.

- Very recently, in 2009, Jaszkiewicz and Zielniewicz [121] adapted PMA (see section 3.4.3 in chapter 3) to the MOTSP. They also analyzed the idea of path relinking (search for new efficient solutions along path in the decision space connecting two potentially efficient solutions). Whenever two parents are selected for recombination, a path in the decision space linking the two solutions is built by the local search that starts from one of the parents. The local search uses lexicographic optimization: first it minimizes the distance (measured by the number of different arcs) to the second parent. In case of several solutions having the same distance they are compared with the current scalarizing function. This adaptation of PMA with path-relinking is further extended with the use of one iteration of PLS (see section 3.4.1 in chapter 3). Both PLS and path-relinking use a simple two-edge exchange neighborhood. Moreover, they used for the generation of the initial population and for improving the solutions after recombination a greedy version of the Lin-Kernighan algorithm [152]. The results of PMA improve those of PD-TPLS [187] (see next section) on the biobjective Kro instances of 100 or 200 cities, but with higher running time and higher number of parameters.

5.2 Local search algorithms

Another class of metaheuristics abundantly applied to the MOTSP is sophisticated local search algorithms. The two first local search presented are based on the Pareto local search method (see section 3.4.1 in chapter 3).

- Paquete and Stützle [187] presented in 2003 three versions of a local search method, all presenting two phases and applied to the biobjective TSP. The methods are called two-phase LS (TPLS), double two-phase LS (P-TPLS) and Pareto double two-phase LS (PD-TPLS). The two phases are as follows. In the first phase, the TPLS method generates a single initial solution by optimizing only the first objective (f_1) with a LS (the weight associated to f_1 is equal to one and the weight associated to f_2 is equal to zero). In the second phase, the LS is applied with a sequence of different weighted sum a_i forming a chain. The LS with the function a_{i+1} starts with the best solution found at the preceding iteration i, the function a_{i+1} is a slight modification of a_i: the weight associated to f_1 is decremented and the weight associated to f_2 is incremented.

  P-TPLS is simply TPLS applied two times, considering for the first phase respectively f_1 and f_2.

  PD-TPLS tries to determine additional potentially non-dominated points which are missed by the aggregation used. For this, a LS that accepts solutions which are not dominated by the current local optimum in both objectives, is applied after a local optimum is found with each function a_i. The neighborhood used is the two-edge exchange.

  A comparison of the results obtained shows that they are better than those of MOGLS on the Kro biobjective instances of 100 cities.

- In 2004, Angel *et al.* [6] considered a so-called "dynasearch neighborhood". Given a solution δ and a neighborhood $\mathcal{N}(\delta)$, the first idea is to determine the set $U\mathcal{N}(\delta)$ containing all neighboring solutions not dominated by δ. The authors applied this idea to the particular dynasearch two-edge exchange neighborhood, called $\mathcal{N}_{dyna}(\delta)$, which consists in applying a series of independent two-edge exchange moves: two such moves are independent if the edges involved do not overlap. They use a dynamic programming method to compute $U\mathcal{N}_{dyna}(\delta)$ in an efficient way.

  The neighborhood is integrated in the first version of PLS (PLS1, see section 3.4.1 in chapter 3). The initial population is composed of only one random solution. The authors also used a rounding technique to limit the number of potentially non-dominated points generated. The results on the biobjective Kro instances of 100 cities are of good quality but obtained in a high resolution time.

- In 2004, Paquete *et al.* [184] applied the second version of PLS (PLS2, see section 3.4.1 in chapter 3). Three neighborhoods are considered: two-edge exchange, a so-called 2h-opt (considering in addition of two-edge exchange moves, moves of a single city from its position to another inside the tour) and three-edge exchange moves. With this last neighborhood, the results are a little bit better than those of MOGLS on the Kro instances of 100 cities but at the price of a higher computation time.

- A quite different approach is proposed by Li [150] in 2005, based on the notion of "attractor" of each objective. Such attractor contains a set of the best solutions found for the corresponding objective. The number of hits of each edge in all these solutions is recorded. Using this information, the procedure combines these edges to produce potentially efficient solutions. Results are provided for randomly generated costs, but no comparison with other algorithms, neither assessment of the quality of the approximations is realized.

- In 2009, Paquete and Stützle [189] studied the working mechanisms of MOMHs, and in particular for tackling the MOTSP. They considered the so-called scalarized acceptance criterion (SAC) search model, that is methods that are based on the scalarization of the objectives. They analyzed the number of scalarizations, the search strategy followed, the computation time invested for tackling each of the resulting single-objective problems and various other components. At final, they obtained a method, very close to the TPLS method that they proposed earlier. They used different types of instances: Euclidean instances, random instances and mixed instances, with two or three objectives and including 100, 300 or 500 cities. This is the first time than instances of such size (500 cities) were tackled. They compared the performances of their approach with MOGLS, in a very accurate way, and obtained better results.

5.3 Ant colony optimization

Another more recent and quite different metaheuristic is ant colony optimization (ACO) [48, 49] which also has been adapted to a multiobjective framework.

- In 2007, Garcia-Martinez *et al.* [79] authors reviewed and classified the different existing MOACO algorithms by proposing a taxonomy and developing a large and systematic experimental study, comparing them with SPEA2 and NSGA-II (see section 3.4.2 in chapter 3) on several MOTSP instances, for the Kro instances of 50 and 100 cities.

  From this paper, it results that MOACO algorithms are very competitive against those two MOGAs. Among their conclusions, the authors proposed as future development to add a local search to the MOACO algorithms with the aim to compare their performances with those of MOGLS.

- Very recently, in 2009, López-Ibáñez and Stützle [154] studied the components of MOACO algorithms, like the use of local search, the use of one versus several pheromone matrices, or the use of one or several ant colonies. They applied their analysis to Euclidean biobjective instances of 500 cities, with different correlations.

5.4 Particular studies and variants of the MOTSP

To be as far as possible exhaustive, we mention here other papers related to the MOTSP.

- Gupta and Warburton [95] described in 1986 a heuristic method to approximate an optimal solution minimizing a weighted Tchebycheff scalarizing function.

- Tung [224] proposed in 1994 an exact labeling method to determine X_E. Of course, only very small instances can be treated.

- Special models are also treated by different authors:
  - Fisher and Richter [71] considered in 1982 a biobjective particular TSP, in which the first objective is a classical sum objective while the second is a multiplicative reliability objective, and they proposed a dynamic programming approach.
  - In the biobjective problem treated by Melamed and Sigal [170] in 1997, the first objective is also a classical sum objective while the second is the maximum of the cost of the edges selected for the tour; their method is based on the linear convolution of the two objective functions.
  - The TSP model treated by Angel *et al.* [5] in 2004 is called TSP(1,2) because the cost associated to each edge can only take two values: 1 or 2. For a biobjective TSP(1,2), the authors proposed a LS based on the two-edge exchange neighborhood to obtain an approximation $\widetilde{Y}_N$ with a guarantee of performance equal to 3/2.
  - Manthey and Ram [167] also described in 2006 heuristics with guarantee of performance in the particular case where the costs of the MOTSP satisfy the $\gamma-$triangle inequality:

    $$c_k(i,j) \leq \gamma\big(c_k(i,l) + c_k(l,j)\big) \quad \forall\, i,j,l \in \{1,\ldots,n\}, \forall\, k \in \{1,\ldots,p\}, \gamma \in [\frac{1}{2}, 1].$$

    In this paper, they also tackled the case of MOTSP(1,2).
  - In 2008, Ozpeynirci [180] studied in his thesis two special cases of the MOTSP. In both cases, additional constraints are considered. In the first case, a set of constraints is defined on the distances between cities, such that the optimal tour has a special structure, that is, it looks like a pyramid when the numbers of the cities are plotted in the order they are visited by the optimal tour. In the second case, the complete graph associated to the MOTSP has a special structure (Halin graphs) such that only some roads between cities are available. He proposed exact algorithms for both cases.

Some variants of the MOTSP are also considered.

- An interesting variant of the TSP is the TSP with profits. In this problem, the traveling salesman is not required to visit all cities, but a profit is associated to each visited city. It can thus be modeled as a biobjective problem: it is necessary to find a cyclic permutation over a subset of the n cities such that the collected prize is maximized while the travel cost is minimized. The old paper of Keller and Goodchild [132] from 1988 already introduced such a model for which a heuristic is proposed. Very recently, in 2009, Bérubé *et al.* [81] proposed an exact ϵ-constraint method to generate X_E and Jozefowiez *et al.* [128] in 2008 developed a MOGA combined with an ejection chain local search to approximate X_E.
- Another interesting variant of the TSP is the TSP with sequence priorities [207], introduced by Schmitz and Niemann. In this problem, the first objective is the traditional cost associated to a tour, while the second objective is the distance between the tour and a sequence that the tour has to respect, due to arrival times of delivery requests. They applied for this particular problem a local search integrating different specific means of intensification and diversification.
- Other related models are analyzed: Tuyttens *et al.* [226] in 2004 and Jozefowiez *et al.* [129] in 2008 considered the multiobjective vehicle routing problems and Huang *et al.* [107] in 2006 a multiobjective route planning problem.

6
New methods of resolution

In this chapter, we present the three new methods of resolution that have been developed in this thesis. The first method is a direct adaptation of tabu search to MOO (section 6.1), called Pareto ranking tabu search (PRTS). We present then the memetic algorithm MEMOTS that integrates this tabu search (section 6.2). The last method presented (section 6.3), the two-phase Pareto local search (2PPLS), is rather different and follows the exact two-phase method but heuristic methods are used in both phases.

6.1 Pareto ranking tabu search (PRTS)

We present in this section the Pareto ranking tabu search (PRTS) method. The PRTS method follows the MOLS2 model presented in section 3.4.1, chapter 3: the method is started from only one solution, an initial solution $x^0 \in \mathcal{X}$ and at each iteration, non-tabu neighbors are generated from the current solution x^c. The best neighbor is then selected and becomes the next current solution x^c. To evaluate the neighbors in a MO context, we employ an original evaluation function of the neighbors, based on a recent paper of Elaoud et al. [62]. In this paper, a genetic algorithm (called PFGA for Pareto fitness genetic algorithm) using a new fitness function based on individuals rank and density, is developed. This approach giving promising results, the fitness function of PFGA has been adapted within a tabu search for the evaluation and the selection of the neighbors.

Two measures are considered in the evaluation function of PFGA: a classification of the individuals according to an original rank, called double Pareto ranking (*DPR*), and a division of the objective space in hypervolumes allowing to measure the individuals density. These two concepts are then aggregated into a single evaluation function.

If PRTS is used as a LS in a memetic algorithm, its role is to intensify the search. Therefore, it is preferable to use only the rank for the selection of the best neighbor. However, if the tabu algorithm is used in an independent way, that is without integrating it in a population-based method, it is preferable to use the density measure in the neighbors evaluation since diversification and intensification are searched out at the same time. The addition of the density measure gives the so-called PRTS+D method, which has already been exposed in previous works [158, 160, 161].

We only present and use in this thesis the PRTS method, based on *DPR*.

The *DPR* of a neighbor i is determined in two times. Initially, the Pareto ranking (*PR*) is computed by counting the number of individuals of the population P which dominate the neighbor:

$$PR(i) = \Big|\{j | j \in P, f(j) \prec f(i)\}\Big|,$$

In PRTS, the considered population includes the neighbors and the potentially efficient solutions found so far by the algorithm.

The *DPR* of a neighbor i is then defined by the sum between its $PR(i)$ and the $PR(j)$ of its

dominators j:

$$DPR(i) = PR(i) + \sum_{j \in P,\, f(j) \prec f(i)} PR(j)$$

By way of example, the values of PR and DPR for a population of 13 individuals are presented in Figure 6.1. If we compare the individuals I_1 (of PR and DPR respectively equal to 4 and 5) and I_2 (with PR and DPR respectively equal to 4 and 9), we note that their PRs are equal since each one is dominated by four individuals. However, the individual I_1 seems more interesting since the individuals that dominate it have a PR equal to 0 or 1, while the PRs of the individuals that dominate the individual I_2 are equal to 0, 1 or 3.

Thus, by using DPR, the individual I_1 will be preferred since the PRs of the individuals that dominate it are lower, which results in a lower DPR.

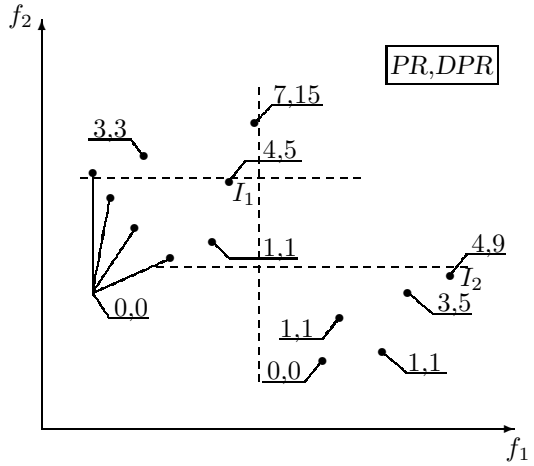

Figure 6.1: Illustration of PR and DPR.

After the best non-tabu neighbor selection, the set of potentially efficient solutions $\widetilde{X}_E$ is improved by adding the neighbors presenting a DPR value equal to 0 and different in the objective space than the solutions of $\widetilde{X}_E$ (only an approximation of a minimal complete set is generated). In addition, the set $\widetilde{X}_E$ is actualized by removing the dominated solutions. This step is done by the `Addsolution` procedure presented in section 3.4 of chapter 3.

The functioning of PRTS is given in Algorithm 14. The method requires an initial solution $x^0 \in \mathcal{X}$, a neighborhood function $\mathcal{N}(x)$ and has two numerical parameters: the tabu tenure t and the number of iterations N.

6.2 Memetic multiobjective tabu search (MEMOTS)

6.2.1 Presentation

The memetic multiobjective tabu search (MEMOTS) follows the MOMA model presented in section 3.4.3, chapter 3. The distinctive feature of MEMOTS in comparison with most of the other MOMAs presented previously in chapter 3, is that scalarizing functions are not used for the parents selection.

Indeed, as illustrated in Figure 6.2, a negative point of this selection is that the distribution of the solutions of the population in objective space is not taken into account; parents being located in a region already well-exploited could be thus selected for the recombination because the weight set λ used to build the scalarizing function on which the selection is based is randomly generated.

Algorithm 14 PRTS

Parameters ↓: an initial solution $x^0 \in \mathcal{X}$, the tabu tenure t, the number of iterations N, a neighborhood function $\mathcal{N}(x)$.
Parameters ↑: An approximation $\widetilde{X}_E$ of the efficient set.

--| Initialization
$i \leftarrow 0$
$\widetilde{X}_E \leftarrow \{x^0\}$
$x^c \leftarrow x^0$
$T \leftarrow \varnothing$
while $i \leq N$ **do**
  --| Generation of the neighbors of x^c
  Generate $\mathcal{N}(x^c)$
  Generation of non-tabu neighbors $y^1, \ldots, y^l, \ldots, y^{|\mathcal{N}(x^c)|}$ of x^c in $\mathcal{N}(x^c)$
  for all $y^l \in \mathcal{N}(x^c)$ **do**
    Computation of $DPR(y^l)$
  Computation of $DPR^* = \min DPR(y^l)$
  Random selection of a neighbor y^* among the neighbors which respect $DPR(y^l) = DPR^*$
  $x^c \leftarrow y^*$
  --| The opposite move is tabu for t iterations
  $T \leftarrow \{x^c \rightarrow y^*\}$
  for all $y^l \in \mathcal{N}(x^c)$ **do**
    if $DPR(y^l) = 0$ **then**
      AddSolution($\widetilde{X}_E \updownarrow$, $y^l \downarrow$, $f(p^l) \downarrow$)
  $i \leftarrow i + 1$

Also, comparing to the MOMA model, only one population is used, the one composed of the potentially efficient solutions. The LS applied to the solution coming from the crossover operator is PRTS.

The functioning of MEMOTS, presented in Algorithm 15, is as follows.

Algorithm 15 MEMOTS

Parameters ↓: an initial population P_0, a crossover operator CO, the maximal number it_{stop} of iterations of PRTS without improvement, the number r of solutions taken into account for the selection of the closest solution to the first parent, the number N of iterations.
Parameters ↑: an approximation $\widetilde{X}_E$ of the efficient set.

--| Initialization
$i \leftarrow 0$
for all $x^e \in P_0$ **do**
  AddSolution($\widetilde{X}_E \updownarrow$, $x^e \downarrow$, $f(x^e) \downarrow$)
while $i \leq N$ **do**
  for all $x^e \in \widetilde{X}_E$ **do**
    Computation of the density $D(x^e)$
  Computation of $D^* = \min D(x^e)$
  Random selection of a parent x^1 among the solutions which respect $D(x^e) = D^*$
  Random selection of a parent x^2 among the $\min(r, |\widetilde{X}_E| - 1)$ solutions of $\widetilde{X}_E$ closest to x^1
  --| Application of the crossover operator CO to x^1 and x^2
  $x^3 \leftarrow CO(x^1, x^2)$
  AddSolution($\widetilde{X}_E \updownarrow$, $x^3 \downarrow$, $f(x^3) \downarrow$)
  Application of PRTS from x^3 until there is no more improvement in $\widetilde{X}_E$ while it_{stop} iterations
  --| We increment the total number N of iterations by the number N_{PRTS} of iterations performed by PRTS
  $i \leftarrow i + N_{PRTS}$

Initially, the set $\widetilde{X}_E$ is updated with each solution x^e belonging to an initial population P_0.

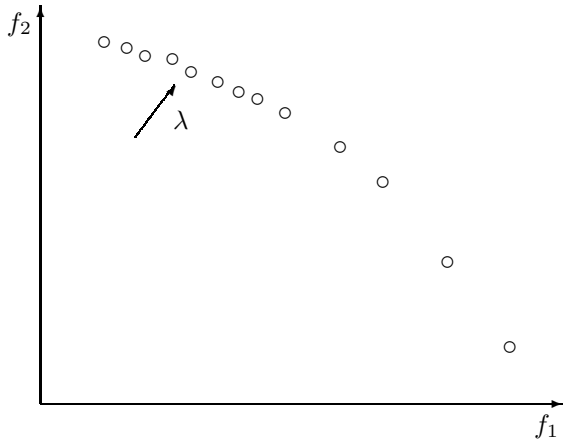

Figure 6.2: Selection based on scalarizing functions.

The initial population should be ideally diversified in the objective space.

Then, two solutions from $\widetilde{X}_E$ are selected (the parents). The first parent, called x^1, is one of the solutions of $\widetilde{X}_E$ having a minimal density, defined by the number of solutions being in the same hypervolume than the solution. The technique employed to realize the division of the objective space in hypervolumes will be explained in section 6.2.2 (this is one of the main differences with respect to other MOMAs).

The second parent, called x^2, is one of the solutions of $\widetilde{X}_E$ among the r closest solutions to the first parent x^1 according to the euclidean distance in objective space. We select randomly a parent x^2 close to x^1 to comply with the conclusions of Ishibuchi who emphasized that it was preferable to combine similar parents in MOO [111]. It should be noted that if the number of potentially efficient solutions generated by the algorithm, noted $|\widetilde{X}_E|$, is lower or equal to r, the selection of the second parent is realized among the $(|\widetilde{X}_E| - 1)$ closest solutions.

Next, both parents x^1 and x^2 are combined with a crossover operator, called CO. The new solution obtained, the offspring, is called x^3. The set $\widetilde{X}_E$ is then updated with this solution. Finally, the PRTS is applied from x^3 until no more improvement in $\widetilde{X}_E$ is observed during a certain iterations number it_{stop}. Finally, this process is iterated by again selecting two parents.

The role of PRTS is to intensify the search by generating new solutions close to the selected parent x^1 of minimal density. The diversification of the method is ensured by the generation of solutions in little or not exploited regions.

An important parameter of MEMOTS is it_{stop}, that is after how many iterations without improvement we have to stop PRTS, knowing that the total number of iterations is limited. It could be preferable to start PRTS quite often from a solution of minimal density rather than strongly intensifying the search in a precise region. The influence of this parameter will be studied later in chapter 7.

6.2.2 The dynamic hypergrid

Another important point not yet explained in this method is how to measure the density of a potentially efficient solution.

As described in section 3.4.2 of chapter 3, two authors have already used the concept of density within the framework of MOO:

- Deb with the concepts of hyperniche [212] and hypercube [46], developed through the NSGA algorithms, for the selection of individuals into a population.
- Knowles and Corne with the hypergrid, introduced into the (1+1)-PAES method [136], employed in the rule of acceptance of a new solution and also used to limit the number of

solutions generated. The number of divisions of the hypergrid is fixed at the beginning of the algorithm.

These techniques have then been used in other algorithms, for instance, Elaoud *et al.* [62] also used an hypergrid to measure the density of individuals. In this case, the number of divisions of the hypergrid depends on the number of individuals.

An inconvenience of the hyperniche and hypercube measures is the high computational time, because each solution has to be compared to all others to obtain the density of all solutions. Another argument in discredit of the hyperniche measure is that this measure is extremely dependent on the size of the niche [45].

Consequently, we use a hypergrid in MEMOTS, created by a division of the objective space in hypervolumes. The density of a solution is thus defined by the number of solutions being in the same hypervolume than the solution. In this way, it is quite easy to compute the density of a solution by always having in memory the number of solutions in each hypervolume and the coordinates of the solutions in the hypergrid. Thus, if we want to know the density of a solution, it is enough to read the number of solutions present in the hypervolume identified by the coordinates of the solution.

The size of the hypergrid has to be managed. Indeed, if the number of hypervolumes that compose the hypergrid is too high, all solutions will be in different hypervolumes. On the other hand, if there are not enough hypervolumes, a lot of solutions will be located in the same hypervolumes (see Figure 6.3).

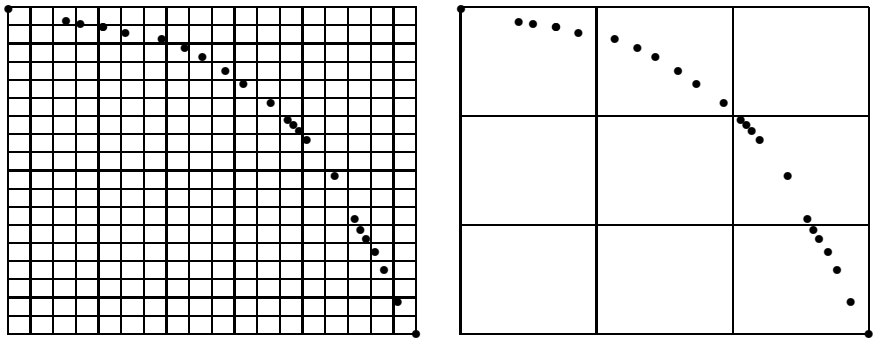

Figure 6.3: On the left: too many divisions, a high number of solutions have a density equal to one. On the right: not enough divisions, a solution of minimal density cannot be distinguished.

In these two cases, it will be difficult to distinguish a solution of minimal density. This is why the size of the hypergrid is dynamically updated. As the method starts from a low number of solutions and this number is, for the majority of the cases, in constant increases when a MOP is solved, the number of divisions of the hypergrid is also constantly increased. The rule is as follows: if the average density of the solutions becomes higher than a certain threshold, the number of divisions of the hypergrid is increased by a certain step. In this way, a good measure of density is guaranteed. It should be noted that the average density is computed by only considering the hypervolumes containing at least one solution.

The illustration of this technique is represented in Figure 6.4, where the evolution of the number of divisions of the hypergrid according to the iterations number is represented.

The size of the hypergrid is also updated as soon as a new solution being outside of the current hypergrid is produced. Between each update, we keep in memory the hypergrid and the coordinates of the solutions, which makes it possible to reduce the computational time but however, increases the requirement in memory space.

This way of managing the hypergrid implies to set the mean density threshold from which the number of divisions of the hypergrid is increased. The influence of this parameter is experimented in section 7.5.2 of chapter 7. It will be shown that the value of this parameter remains relatively

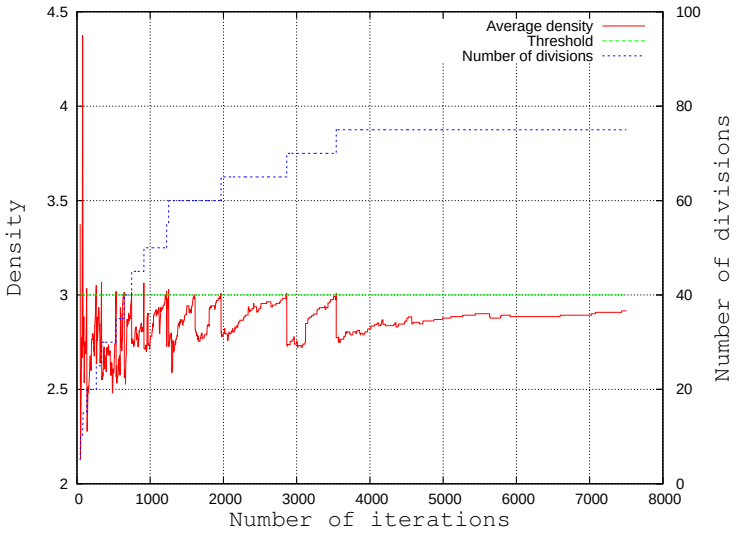

Figure 6.4: Evolution of the number of divisions of the hypergrid. Each time that the average density becomes higher than the density threshold, the number of divisions of the hypergrid is increased (these results come from the application of MEMOTS to the MOMKP, see chapter 7).

easy to set, since the sensitivity of the algorithm with respect to the value taken by this parameter is not strong.

6.3 Two-phase Pareto local search (2PPLS)

The two-phase Pareto local search (2PPLS) is a heuristic adaptation of the exact two-phase method developed by Ulungu and Teghem [231] presented in section 3.2.3 of chapter 3. The two phases of the method are as follows:

1. Phase 1: generation of a good approximation of a minimal complete set of extreme supported efficient solutions, called $\widetilde{X}_{SE1_m}$. These solutions can be obtained by solving weighted sum single-objective problems obtained by applying a linear aggregation of the objectives.

2. Phase 2: generation of a good approximation of the non-supported efficient solutions located between the supported efficient solutions. In this phase, we apply the Pareto local search (PLS) method [6, 185], presented in section 3.4.1 of chapter 3.

It should be noted that 2PPLS is close to the method of Hamacher and Ruhe [98], developed in 1994, to find a well-distributed approximation of the efficient solutions of the MO spanning tree problem. However, two differences can be noticed: they use the exact set of extreme efficient solutions in phase 1 and a stop criterion related to the distance between two consecutive solutions is applied in phase 2. As far as we know, this method has only been applied to the MO spanning tree problem.

6.3.1 First phase: generation of the initial population $\widetilde{X}_{SE1_m}$

To generate an approximation $\widetilde{X}_{SE1_m}$ of the extreme supported efficient solutions, we follow the same dichotomic scheme than in Algorithm 3 presented in section 3.2.3 of chapter 3. Consequently, the first phase can only be adapted to solve biobjective problems. As a reminder, this dichotomic method consists in generating all weight sets which make it possible to generate all

supported efficient solutions (the set X_{SE1}) by solving weighted sum problems, for a biobjective problem. The main difference is that we use a heuristic method to solve the different weighted sum problems, as the use of an exact method to solve the single-objective problems can be very time-consuming for $\mathcal{NP}$-Hard problems. As a result, the solutions obtained are not necessarily efficient, nor supported efficient but constitute a set of solutions that can be very close to X_{SE1_m}, depending on the quality of the heuristic used to solve the weighted sum problems.

In addition, two simplifications in comparison with the exact algorithm are considered:

- A search for the lexicographic solutions is not realized. We are satisfied with the solutions given by the heuristic for the resolution of two single-objective problems with weight sets respectively equal to $(1, 0)$ and $(0, 1)$.
- Because the values taken by the weight sets given by the exact method can be very large, we normalize the weight sets such that $\lambda_1 + \lambda_2 = 1$, with $\lambda_1, \lambda_2 \in \mathbb{R}$. In consequence, if we want to use a single-objective solver that only accepts integer data, the new data associated to the weighted sum have to be rounded to the nearest integer value. For instance, for the biobjective TSP, that means that a new matrix of cost C^λ will be generated from the two matrices C_1 and C_2 corresponding to both objectives $\big(C^\lambda = \text{round}(\lambda_1 c^1_{ij} + \lambda_2 c^2_{ij})\big)$. This operation is however only possible if the objectives are of the same type.

The details of this heuristic adaptation are given in Algorithm 16, called `Phase1Heuristic`, which uses Algorithm 17, called `SolveRecursionHeuristic`. The stop criterion of the `Phase1-Heuristic` algorithm is slightly different than the one in the exact Algorithm 3. When we seek for a new solution, to start again the `SolveRecursionHeuristic` algorithm, the solution must be located in the interior of the triangle defined by the points $f(x^r)$ and $f(x^s)$ plus the local ideal point formed by these two points. Indeed, with an exact method, $f(x^t)$ can only be located inside the triangle but with a heuristic method, $f(x^t)$ can also be located outside of this triangle. Considering this case will increase the complexity of the first phase, the running time and the risk of cycling. And as the aim of the first phase is to quickly generate an approximation of the non-dominated supported points, we did not consider this special case and we stop the procedure when $f(x^t)$ is located outside of the triangle.

We note the interior of the triangle by $\text{int} \triangle f(x^r)f(x^s)$. Therefore, if

$$f(x^t) \cap \text{ int} \triangle f(x^r)f(x^s) = \varnothing,$$

the `SolveRecursionHeuristic` algorithm is stopped.

Also, as the new solution x^t can be dominated by other solutions already generated, we use the `AddSolution` procedure to update the set $\widetilde{X}_{SE1_m}$.

Algorithm 16 `Phase1Heuristic`

Parameters $\downarrow$: A biobjective MOCO problem, with $f_1(x)$ and $f_2(x)$ both objectives.
Parameters $\uparrow$: An approximation $\widetilde{X}_{SE1_m}$ of X_{SE1_m}.

--| Heuristic resolution of the single-objective version of the MOCO problem by only considering the first objective f_1
`SolveHeuristic` $\min_{x \in \mathcal{X}} \big(f_1(x)\big)$
Let x^1 the solution returned by the heuristic
--| Heuristic resolution of the single-objective version of the MOCO problem by only considering the second objective f_2
`SolveHeuristic` $\min_{x \in \mathcal{X}} \big(f_2(x)\big)$
Let x^2 the solution returned by the heuristic
--| Computation of an approximation $\widetilde{X}_{SE1_m}$ of X_{SE1_m}
`SolveRecursionHeuristic`$(x^1 \downarrow, x^2 \downarrow, \widetilde{X}_{SE1_m} \updownarrow)$

Algorithm 17 `SolveRecursionHeuristic`

Parameters $\downarrow$: x^r, x^s.
Parameters $\updownarrow$: $\widetilde{X}_{SE1_m}$.

--| Creation of a weight set λ normal to the line segment connecting $f(x^r)$ and $f(x^s)$
$\lambda_1 = \frac{f_2(x^r)-f_2(x^s)}{f_2(x^r)-f_2(x^s)+f_1(x^s)-f_1(x^r)}, \lambda_2 = 1-\lambda_1$

--| Resolution of the single-objective version of the MOCO problem by considering the weighted sum
`SolveHeuristic` $\min_{x\in\mathcal{X}}\big(\lambda_1 f_1(x) + \lambda_2 f_2(x)\big)$
Let x^t the solution returned by the heuristic
--| Add x^t to $\widetilde{X}_{SE1_m}$
`AddSolution`$(\widetilde{X}_{SE1_m}\ \updownarrow,\ x^t\ \downarrow,\ f(x^t)\ \downarrow)$

if $f(x^t) \cap \text{ int} \triangle f(x^r)f(x^s) \neq \varnothing$ **then**
  `SolveRecursionHeuristic`$(x^r\ \downarrow, x^t\ \downarrow, \widetilde{X}_{SE1_m}\ \updownarrow)$
  `SolveRecursionHeuristic`$(x^t\ \downarrow, x^s\ \downarrow, \widetilde{X}_{SE1_m}\ \updownarrow)$

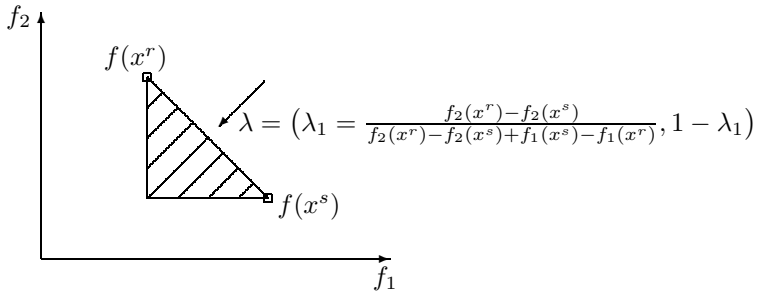

Figure 6.5: Illustration of the stop criterion of the heuristic method for generating an approximation of the extreme supported efficient solutions.

Surprisingly, reducing the running time is not the only advantage related to the use of a heuristic method for solving the weighted sum problems. Indeed, with a heuristic method, non-supported efficient solutions can be generated, which can never happen if an exact solver is employed.

6.3.2 Second phase: improvement of the initial population with PLS

In the exact two-phase method, the second phase is only dedicated to the generation of the non-supported efficient solutions. Here, in the second phase we will explore the neighborhood of the solutions of the initial population, in order to mainly generate solutions located between two consecutive potentially non-dominated points generated during the first phase. With this aim, we use the PLS method [6, 185], as presented in section 3.4.1 of chapter 3. However, the method starts from each solution of the population $\widetilde{X}_{SE1_m}$ generated in phase 1, whereas Angel *et al.* [6] and Paquete *et al.* [185] start PLS from a randomly generated solution.

The 2PPLS method stops when the second phase stops, that is when a minimal critical non-dominated Pareto local optimum set has been found, with respect to the neighborhood considered in PLS. No numerical parameter is thus needed to define a stop condition.

7 Resolution of the MOMKP

In this chapter, we apply PRTS, MEMOTS and 2PPLS with the aim of solving the multidimensional multiobjective knapsack problem (MOMKP). We first present, for each of the methods, how they have been adapted in order to (heuristically) solve the MOMKP (sections 7.1, 7.2 and 7.3). The data sets are described in section 7.4 and the results of an intensive study of the influence of the values of the parameters of the methods are then showed (section 7.5). Finally, we compare the results of MEMOTS and 2PPLS (section 7.7) between them and the results of the methods with state-of-the-art results (section 7.8).

As a reminder, the MOMKP is defined as follows:

$$\begin{aligned} \text{“max”} \quad & f_k(x) = \sum_{i=1}^{n} c_k^i x_i \qquad k = 1, \ldots, p \\ \text{subject to} \quad & \sum_{i=1}^{n} w_j^i x_i \leq W_j \qquad j = 1, \ldots, m \\ & x_i \in \{0, 1\} \qquad i = 1, \ldots, n \end{aligned} \tag{7.1}$$

In all methods, the coding of a solution is done with the classic binary coding.

7.1 Adaptation of PRTS to the MOMKP

To adapt the Pareto ranking tabu search (PRTS, see chapter 6, section 6.1) to the MOMKP, we have to define the initial solution, the neighborhood and the management of the tabu list.

7.1.1 Initial solution

The initial solution is randomly generated. Each item has an arbitrary probability of 1/4 to enter in the knapsack. The procedure is stopped when the knapsack is full or when all items have been considered. The quality of the initial solution is a moot point, since the first application of the neighborhood will substantially improve its quality.

7.1.2 Neighborhood

The neighborhood $\mathcal{N}(x)$ is defined in the following way:

- Random generation of a weight set λ.
- Creation of a list containing the items s present in the current solution x (identified by $x_s = 1$) sorting in the increasing order of the following ratio (R_1):

$$R_1 = \frac{\sum_{k=1}^{p} \lambda_k c_k^s}{\sum_{j=1}^{m} w_j^s} \tag{7.2}$$

The items put at the beginning of the list are thus the items of low ratio profit/weight. The size of the list is limited at β percent of n_o, with n_o equal to the number of items present in the solution x.

- Generation of the neighbors:
  - Removing of a combination of q items from x by randomly selecting them from the list. The binary variables x_i associated to the removed items i are put to 0.
  - Filling to the maximum the knapsack with the items t not in the solution x (identified by $x_t = 0$) and not removed at the previous step, taken in the decreasing order of the following ratio (R_2):

$$R_2 = \frac{\sum_{k=1}^{p} \lambda_k c_k^t}{\sum_{j=1}^{m} \left(\frac{w_j^t}{W_j - \sum_{i=1}^{n} w_j^i x_i} + 1 \right)} \tag{7.3}$$

    This procedure is also used in MOGLS [119], and consists in filling the knapsack with the items of high profit and low weight, by giving higher influence to the constraints the load of which (measured by $\sum_{i=1}^{n} w_j^i x_i$) is close to the maximum capacity W_j.

Therefore, roughly speaking, the neighbors are generated by removing the "worst" items from the current solution and by adding the "best" items not in the current solution. We fix the number of neighbors generated at each iteration to the minimum between $\beta \cdot n_o$ (size of the list) and $C_q^{\beta \cdot n_o}$ (number of combinations of items that can be removed). If the number q of items that have to be removed is higher than $\beta \cdot n_o$, we only remove $\beta \cdot n_o$ items.

The neighborhood parameters are thus β, the percentage of items present in the solution x and candidates to be removed and q, the number of items removed.

The movement can be thus characterized by an array giving the items removed and the items added.

7.1.3 Tabu list management

At each movement $x \leftarrow y^*$ (y^* is the best neighbor), we record only a part of the movement attribute, namely the array giving the items removed. The selection of any of these items to fill the knapsack is then forbidden for the t (tabu tenure) next iterations.

7.2 Adpatation of MEMOTS to the MOMKP

To adapt MEMOTS (see chapter 6, section 6.2) to the MOMKP, we need to define the crossover operator, the local search and the way to generate the initial population.

7.2.1 Crossover operator

The crossover operator used is a simple one-point operator starting from both parents x^1 and x^2. As this operation can produce two solutions, we only consider the solution having the left part of x^1 and the right part of x^2. The new solution obtained is called x^3.

With this crossover, the very infrequent case where the solution x^3 does not contain any items can occur. In this case, we cancel the crossover and x^3 is simply equal to x^1.

A more frequent case is when the solution x^3 does not respect the constraints of the problem. We thus apply a repair procedure, inspired by the Michalewicz and Arabas procedure [173] for

the single-objective MKP, which consists in removing the items s in the increasing order of the following ratio (R_3):

$$R_3 = \frac{\sum_{k=1}^{p} \lambda_k c_k^s}{\sum_{j=1}^{m} \left(w_j^s \cdot \max\left(0, \sum_{i=1}^{n} w_j^i x_i - W_j\right)\right)} \tag{7.4}$$

until satisfaction of the constraints.

For an item s, this ratio corresponds to a linear aggregation of the profits generated by the item on the sum of the weights of the item multiplied by the value of the violation of the constraints (if there is no violation in the constraint j, the weight w_j^s of the item s is not considered in the sum). This procedure is also used in the MOGLS method [119].

In MOGLS, the weight set λ is the same weight set as the one employed to build the scalarizing function used for the selection of x^1 and x^2. On the other hand, in MEMOTS, no scalarizing function is used in the selection process. A random weight set could generate a solution x^3 located quite far from x^1 and x^2, and thus not anymore located in a region of minimal density.

Therefore, the weight set is determined according to the potentially efficient solution x^1 selected, in order to guarantee that the better the evaluation of the solution x^1 according to an objective, the higher the value of the weight according to this objective is. The computation formula of the weight set λ is thus the following one:

$$\lambda_k = \frac{N\left(k, f_k(x^1), \widetilde{X}_E\right)}{\sum_{k=1}^{p} N\left(k, f_k(x^1), \widetilde{X}_E\right)} \quad k = 1, \ldots, p \tag{7.5}$$

where the function $N\left(k, f_k(x^1), \widetilde{X}_E\right)$ gives the number of potentially efficient solutions in $\widetilde{X}_E$ whose evaluation according to the objective k is lower than $f_k(x^1)$. The value of this function is ranged between 0 and $\left(|\widetilde{X}_E| - 1\right)$, where $|\widetilde{X}_E|$ represents the number of potentially efficient solutions generated.

In this way, we obtain a weight set for the repair procedure which goes in the direction of the minimal density parent x^1.

7.2.2 Local search PRTS

We use the PRTS method as local search in MEMOTS, with the same adaptation as presented in section 7.1. We however redefine the generation of the weight set λ used in the neighborhood. For the first iteration of PRTS, the weight set λ is the one produced in the crossover operator according to (7.5) (this weight set is thus generated even if the solution x^3 respects the constraints), in order to stay in the region of minimal density during at least one iteration. The weight sets for the next iterations are randomly generated.

7.2.3 Initial population

Two different ways of generating the initial population are experimented:

1. P_1: the initial population is generated by application of PRTS starting from a randomly generated solution. The generation of the initial population is stopped when the number of iterations without improvement is equal to it_{stop} and when at least $r + 1$ potentially non-dominated solutions have been generated.

2. P_2: the initial population is generated exactly in the same way as in the MOMHLIB library of Jaszkiewicz [113] (we have used this library to generate the initial population).

To create a new solution, the items are added to the knapsack one by one. At each iteration, the item s that maximizes the following ratio (R_4) is selected:

$$R_4 = \frac{\sum_{k=1}^{p} \lambda_k c_k^s}{\sum_{j=1}^{m} \left(\frac{w_j^s}{W_j - \sum_{i=1}^{n} w_j^i x_i + 1} \right)} \tag{7.6}$$

The weight set is randomly generated.

As in [118], this procedure is stopped when the average quality, over all utility functions, of the average quality of the K best solutions for each utility function is the same as the quality of local optima of these functions. This stop criterion determines the number of utility functions used: as long as the stop condition is not met, new utility functions are created.

The advantage of this method is that we do not have to fix a parameter equal to the number of initial solutions to generate, while guaranteeing an initial population of good quality.

7.3 Adaptation of 2PPLS to the MOMKP

The two-phase Pareto local search (2PPLS) method only requires two elements to be adapted to a MOCO problem: an initial population and a neighborhood.

7.3.1 Initial population

In 2PPLS, it is preferable to start the method from an initial population of good quality, and ideally the supported efficient solutions. The dichotomic method of Aneja and Nair coupled with an exact single-objective solver (see chapter 3, section 3.2.3) can be used to produce the supported efficient solutions of biobjective instances, even if this possibility is very time-consuming. Another possibility is to use the heuristic adaptation of the method of Aneja and Nair (see chapter 6, section 6.3.1). In this case, a good single-objective heuristic for the MKP has to be employed. Unfortunately, it is not as easy than with the TSP (as we will see in the next chapter) to find an implementation of an efficient heuristic for the MKP. Therefore, we will simply use the greedy method presented in section 7.2.3, with the ratio R_4 (defined by (7.6)) as single-objective heuristic. The different possibilities for the generation of initial population are then the following:

1. P_1: This population is composed of extreme supported efficient solutions, generated with the exact procedure of Aneja and Nair coupled with the free exact LP_solve solver (only for the biobjective instances).

2. P_2: Same initial population than P_2 presented in section 7.2.3.

3. P_3: Use of the greedy heuristic (with the ratio R_4, defined by (7.6)) and with a number of weight sets equal to a parameter S. The weight sets are uniformly distributed, and generated as follows (only for the biobjective instances):

$$\Big((1,0),(1-\frac{1}{S-1},\frac{1}{S-1}),(1-\frac{2}{S-1},\frac{2}{S-1}),\ldots,(\frac{1}{S-1},1-\frac{1}{S-1}),(0,1)\Big)$$

   If S is equal to 1, the weight set considered is simply (0.5,0.5).

4. P_4: Use of the heuristic adaptation of the Aneja and Nair procedure coupled with the greedy heuristic (only for the biobjective instances).

7.3.2 Neighborhood

The neighborhood in PLS, that is the second phase of 2PPLS, is intensively used (the neighborhood is started from each new potentially efficient solutions contrary to PRTS where the neighborhood is started from only one selected solution) and as consequence has to be simple and fast. We only experiment in this chapter the first version of PLS (PLS1, see Algorithm 9, section 3.4.1 in chapter 3).

Neighborhood $\mathcal{N}_1$

This is definitively the simplest neighborhood that we could define: one item is removed, another is added (1-1 exchange). All the possibilities of 1-1 exchanges are considered (the neighborhood is thus determinist). This neighborhood is quite similar to the two-edge exchange moves for the TSP (see next chapter).

One reproach that we can express about this neighborhood is that once an item has been added, it may remain enough places in the knapsack to add another item. Moreover, with this neighborhood, for each neighbor the number of items present in the knapsack is exactly the same as the number of items present in the current solution. The initial population should thus contain solutions with different number of items (since the efficient set is generally composed of solutions having a different number of items, see [76]).

A way to improve the neighborhood is then the following. If it remains enough places after a 1-1 exchange, the items maximizing the ratio R_2 (defined by (7.3)) are selected to fill the knapsack. To compute the ratio R_2, the weight set is randomly generated. This procedure can do nothing but improving the quality of the neighbors, but on the other hand will increase the computational time of the neighborhood, quite a lot since ratio computations and sorting have to be realized.

This improvement of the $\mathcal{N}_1$ neighborhood will be called $\mathcal{N}_{1+}$.

Neighborhood $\mathcal{N}_2$

One critic about the $\mathcal{N}_1$ neighborhood is that only one item is removed. A natural extension of the 1-1 exchange is the 2-2 exchange. But the number of combinations of two items that can be removed from the current solution can be very high (equal to $C_2^{n_o}$, with n_o equal to the number of items present in the solution) and not manageable. The number of combinations of two items that can enter in the knapsack can also be very high. One possibility to deal with that is to limit the number of items that are candidates to be removed or added as done in section 7.1.2. But this possibility is not very flexible and causes the introduction of new parameters.

Therefore, a more straightforward neighborhood, based on the hybridization (corresponding to the low-level relay hybrid approach, if we refer to the taxonomy of hybrid metaheuristics developed by Talbi [217]) with a method allowing to solve exactly or heuristically small size MOMKP instances, has been defined.

Two lists are created, both of size L: one list (L1) containing the items candidates to be removed (thus present in the current solution) and another list (L2) containing the items candidates to be added (thus missing in the current solution).

To create L1, the ratio R_1 (defined by (7.2)) is used, and to create L2, it is the ratio R_4 (defined by (7.6)). To generate the weight set necessary to the computation of these ratios, we use the same procedure than in section 7.2.1 (Formula (7.5)) except that the comparison is only made inside the population P of PLS1 containing the new potentially efficient solutions, what gives:

$$\lambda_k = \frac{N\big(k, f_k(x^1), P\big)}{\sum_{k=1}^{p} N\big(k, f_k(x^1), P\big)} \quad k = 1, \ldots, p \tag{7.7}$$

Once both lists containing each L items have been created, we merge them to create a new MOMKP instance composed of $(L * 2)$ items. The capacities W_j of the knapsack of the new

instance of the MOMKP created are equal to $W_j - \sum_{\substack{i=1 \\ i \notin L1}}^{n} w_j^i x_i$ with $j = 1, \ldots, m$.

We then have to solve this new MOMKP instance. As the instance is of small size, we can envisage the use of an exact method. Unfortunately, as seen in chapter 4, no efficient exact method has been developed for solving this problem. Therefore, we have implemented a simple enumeration method to generate all efficient solutions. However, the value of L should remain low since otherwise, the small instance could not be solved in a reasonable time. To be able to use a higher value of L, we can also use a heuristic method to solve the small instance. For that, we will employ a simplified version of the MEMOTS method: no hypergrid will be used to select the first parent. The reason is that the number of potentially efficient solutions generated will not be high, and thus managing a hypergrid to select a solution of minimal density cannot be worthwhile. The advantage of not using the hypergrid is the simplification of the method and the elimination of parameters to tune. Consequently, both parents will be selected randomly in the set of potentially efficient solutions. The weight set used in the neighborhood of the PRTS method will be randomly generated.

Once the small instance has been solved, we merge the efficient solutions (or potentially efficient depending on the method used) of the small instance with the current solution, to obtain the neighbors.

7.4 Data and reference sets

As many authors previously did (see chapter 4), we use the MOMKP instances published by Zitzler on `http://www.tik.ee.ethz.ch/~zitzler/testdata.html`. The data sets include 100, 250, 500 and 750 items, with two objectives and two constraints, three objectives and three constraints or four objectives and four constraints. Nevertheless, we do not use the four objectives instances and we will only use the instances of 100 items with only two objectives. It should be noted that, thereafter, when we will speak about, for example, the 250-2 instance, it will mean the instance with 250 items and two objectives.

In order to assess the quality of the approximations generated, we have used the unary indicators presented in chapter 3, section 3.5.

Some of these indicators need a reference set. For the biobjective instances, we use the non-dominated points generated by Tuyttens [225] by applying the ϵ-constraint method coupled with the commercial CPLEX solver. For the three-objective instances, we approximate the efficient solutions by applying several times heuristic methods (the MEMOTS and MOGLS [119] methods) during a high number of iterations. We then only retain the potentially non-dominated points. However, for an instance, the reference set obtained is not exact and a point of the reference set could be found dominated by a point of the algorithms tested in this chapter. That can actually frequently occur since the number of non-dominated solutions of the three-objective instances of the MOMKP tested in this work is very high. The quality indicators of the algorithm would be penalized since in such case the distance between the new potentially non-dominated points and the reference set would be non-null. Therefore, we improve in an artificial way the reference set by adding a certain value ϵ to the evaluation of each solution of the reference set. The value ϵ is determined experimentally: the value must be small but sufficiently large in order to avoid the problem of generation of potentially non-dominated points missing in the reference set. As a result, for the three-objective instances, we will not be able to measure the proportion of non-dominated points generated by an approximation.

To compute the R indicator, we need a reference point y^0 for each of the instances. The values employed are the same as the ones than Jaszkiewicz used in [114]. As also done by Jaszkiewicz in [114], the number of weight sets used to compute R is equal to 201 for the biobjective instances and to 50 for the three-objective instances.

The bounding point considered to compute the hypervolume is simply the point (0,0), for the biobjective instances. For the three-objective instances, we did not compute the hypervolume since its computation was very time-consuming given the high size of the sets.

In Table 7.1, we show the values of the hypervolume $\mathcal{H}$ and R indicator obtained by the Pareto front for the biobjective instances, as well as the cardinality of the Pareto front ($|Y_N|$), the cardinality of the set containing the supported non-dominated points ($|Y_{SN}|$) and the cardinality of the set containing the non-supported non-dominated points ($|Y_{NN}|$). We do not indicate the values of the distance D_1 and D_2 since these values are equal to 0 for the Pareto front, as well as the value of $I_{\epsilon 1}$ which is equal to 1.

We note that the number of non-supported non-dominated points strongly evolves and is much higher than the number of supported non-dominated points.

In Table 7.2, we show the values of the R and $|RS|$ (cardinality of the reference set) indicators obtained by the references sets for the three-objective instances (the values of D_1 and D_2 are equal to zero, and the $I_{\epsilon 1}$ value to 1).

Table 7.1: Values of the $\mathcal{H}$, R, $|Y_N|$, $|Y_{SN}|$ and $|Y_{NN}|$ indicators obtained by the exact non-dominated points set for the biobjective instances.

Instance	Set	$\mathcal{H}(10^7)$	R	$\lvert Y_N\rvert$	$\lvert Y_{SN}\rvert$	$\lvert Y_{NN}\rvert$
100-2	Pareto front	1.7004	103.084527	121	17	104
250-2	Pareto front	9.8711	245.601700	568	40	528
500-2	Pareto front	40.7941	430.618224	1416	77	1339
750-2	Pareto front	89.3641	741.732968	3030	109	2921

Table 7.2: Values of the R and $|RS|$ indicators obtained by the reference sets for the three-objective instances.

Instance	Set	R	$\lvert RS\rvert$
250-3	Reference Set	256.348239	12419
500-3	Reference Set	509.108855	23318
750-3	Reference Set	747.157385	31523

7.5 Study of the influence of the parameters of the methods

In this large section[1], we carry out an intensive study of the influence of the different parameters of the methods. To do that, we generally represent figures representing the evolution of indicators according to the values of the parameters or to the running time. When it does not reduce the quality of the representations, we represent the data in the form of 'Box-and-Whisker Plot', allowing to emphasize the distributions of the variables studied. The ends of the box represent the first and third quartiles of the distribution and the horizontal line in the box gives the median. We also add segments at the ends of the box representing the extreme values of the distribution. When it will be necessary, colors will be used in the figures.

The computer used for the experiments is a Pentium IV with 3 Ghz CPUs and 512 MB of RAM. Twenty runs of the algorithms are performed each time. When the use of the box will not be possible, the values of the indicators on the figure will correspond to the average value over the twenty runs. The running time of our implementation of the algorithms corresponds to the wall clock time.

7.5.1 Study of the influence of the parameters of PRTS

PRTS is a very basic method, which is used as local search in MEMOTS. Using this method independently can only be efficient on small size problems. This is why we test the influence

[1]This section is not an essential part of this work and we advise the reader who wants to get straight to the point, to bypass this section and to directly go to section 7.6.

of the tabu tenure and the parameters of the neighborhood only on the small size biobjective 100-2 instance. The parameters of PRTS are exposed in Figure 7.1, that is the number N of iterations, the tabu tenure t and the parameters β and q of the neighborhood. We study the influence of t, β and q. To realize these experiments, the basic values (that is the values that remain constant when one parameter is modified) of the parameters are the following: N is fixed to 7500, t to 7 and β and q are respectively equal to 10 and 1.

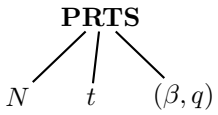

Figure 7.1: Parameters of PRTS.

Influence of the tabu tenure t

We vary the value of t from 0 to 30. We measure the evolution of the D_1 and D_2 indicators according to t. The results are shown in Figure 7.2. We can see that the tabu tenure t has an impact on the quality of the results and that the optimal value is included between 8 and 12 (for the 100-2 instance).

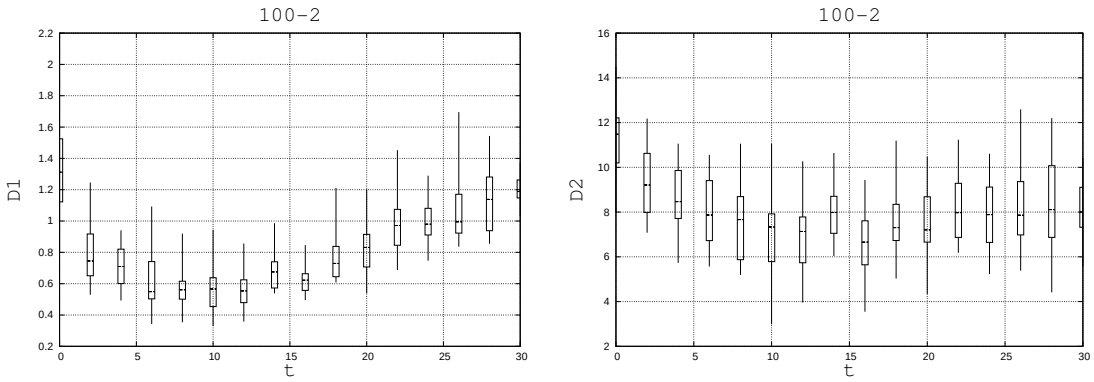

Figure 7.2: Influence of the tabu tenure t of the PRTS method (on D_1 and D_2) for the 100-2 instance.

Influence of the neighborhood parameters

We study here the influence of the parameter β (related to the number of items present in the current solution that can be removed from the current solution) and the parameter q (number of items removed to generate a neighbor). We can see in Figure 7.3 the influences of β and q on D_1 and D_2 for the 100-2 instance. To ensure the visibility of the figures, we only indicate the average value obtained by D_1 and D_2.

For D_1, the best value for q is clearly equal to 1. For D_2, q equal to 2 or 3 gives better values when β is small. On the other hand, when β is increased and $q = 1$, the value of D_2 is decreasing, and with a high value of β, the results for D_2 are equal for $q = 1$ and $q = 2$.

From this small preliminary study, we see that it is more difficult to set parameters of MOMHs in comparison with single-objective metaheuristics, because various contradictory criteria appear in the performances assessment. Consequently, we will always try to use for the parameter values, a compromise between intensification (D_1 minimization) and diversification (D_2 minimization).

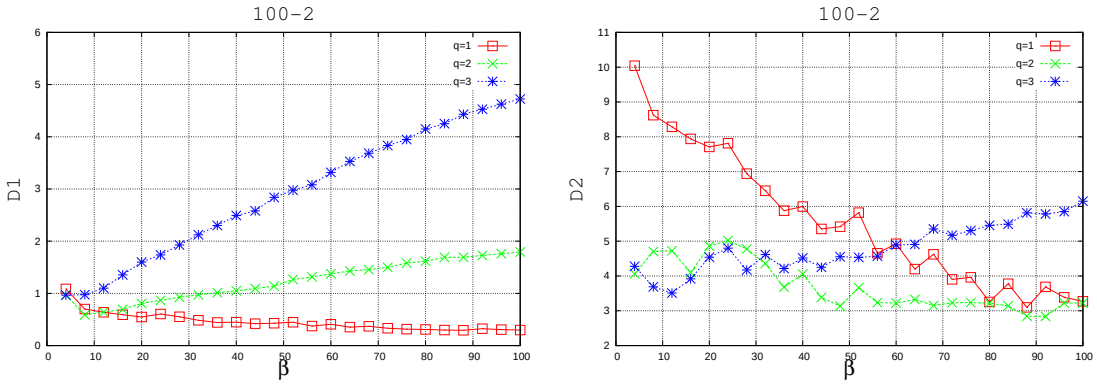

Figure 7.3: Study of the influence of the neighborhood parameters (β, q) on the 100-2 instance for the PRTS method ($\square$: $q = 1$, $\times$: $q = 2$, $*$: $q = 3$).

7.5.2 Study of the influence of the parameters of MEMOTS

MEMOTS presents several parameters as presented in Figure 7.4. Two ways (P_1, P_2) of generating the initial population are possible (see section 7.2.3). We first use the way P_1 to generate the initial population and study the influence of the neighborhood parameters (β, q) of PRTS, the influence of the density threshold *density* from which the number of divisions of the hypergrid is increased by the value taken by the *step* parameter (the starting number of divisions is fixed to 5), the influence of the number it_{stop} of iterations of PRTS without improvement after which PRTS is stopped and the influence of the number r of solutions taken into account for the selection of the closest solution to the first parent. After studying these parameters, we will compare the two different ways of generating the initial population and then we will study the influence of the number N of iterations.

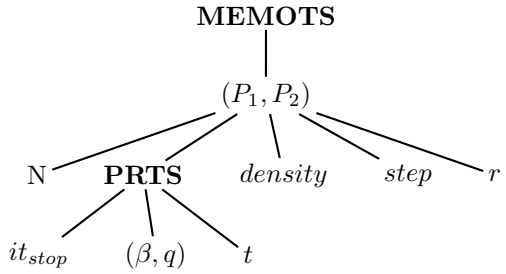

Figure 7.4: Parameters of MEMOTS.

To realize these experiments, the basic values employed for the various parameters are presented in Table 7.3 (it_{stop} equal to 0 means that only one iteration of the tabu search is performed). The influence of the *step* parameter will not be studied and the value of *step* will always be equal to 5.

We start this section by studying the influence of the selection based on the density measure. Then, the influence of the values of the parameters of MEMOTS will be studied.

Influence of the selection based on the density measure

We compare in this section the performances of two ways of selection of the first parent, an important point of the MEMOTS method:

- random selection of the first parent.
- selection based on the density measure.

Table 7.3: Basic values of the parameters of MEMOTS used for the study of the parameters influence.

Instance	N	β	q	$density$	$step$	r	it_{stop}
250-2	7500	10	1	3	5	20	0
500-2	10000	10	1	3	5	20	0
750-2	12500	10	1	3	5	20	0
250-3	10000	10	1	5	5	20	0
500-3	12500	5	2	5	5	20	0
750-3	15000	5	3	5	5	20	0

We have represented in Figure 7.5 the influence of the way to select the first parent on the performances of the algorithm.

The figures represent the evolution of D_1 and D_2 according to the number N of iterations for the 250-3 instance, and for both ways of selecting the first parent. We note that if the selection of the first parent is carried out in a random way, the speed of convergence of D_1 is slightly lower than if the first parent selected is that of minimal density, although the end value is nearly the same. On the other hand, for D_2, convergence is not only slower, but moreover, the final distribution is of worse quality (higher value of the median and higher variations in the distribution). The interest to select the parent of minimal density is thus rather large, but essentially in order to reduce D_2.

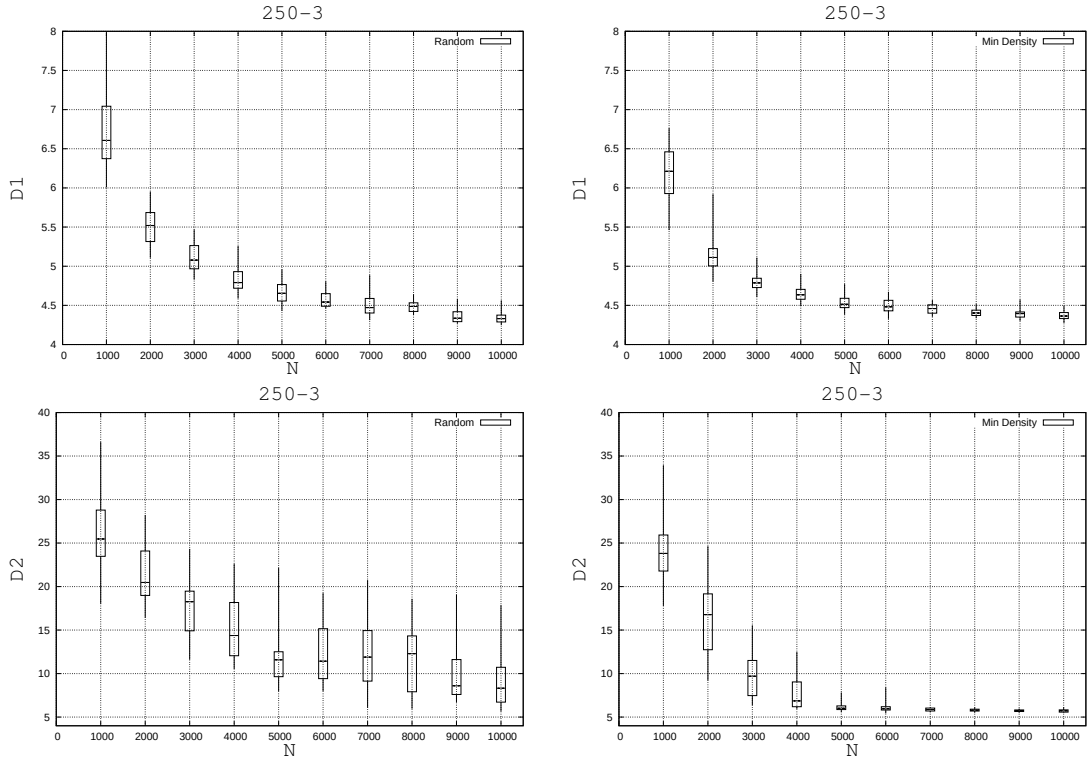

Figure 7.5: Influence of the selection based on the density measure for the 250-3 instance. On the left: random selection. On the right: selection based on the density measure.

Neighborhood parameters

We can see in Figures 7.6 and 7.7 the influences of the β and q parameters on D_1 and D_2 for all instances. To ensure the visibility of the figures, we only indicate the average values obtained by D_1 and D_2.

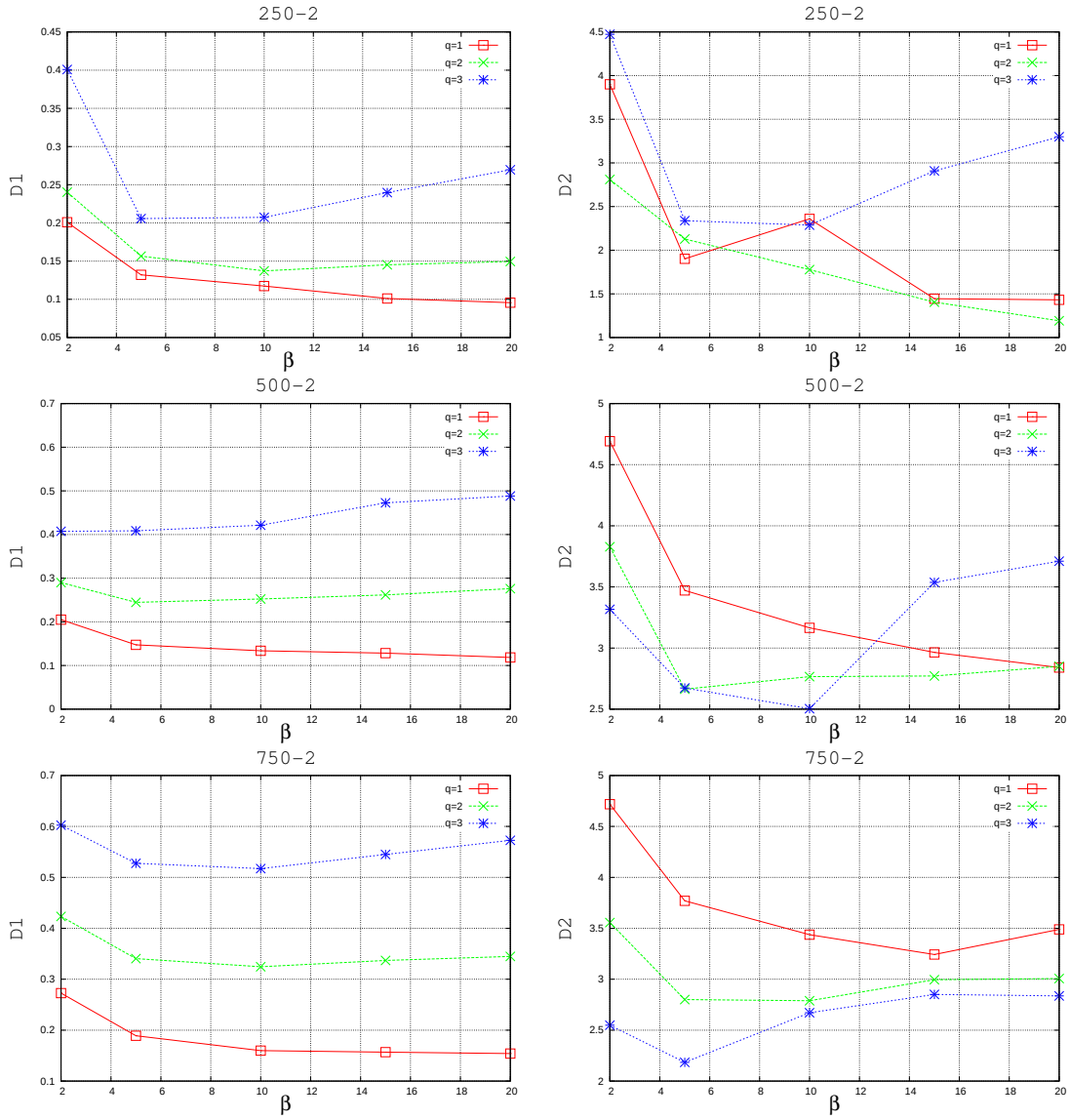

Figure 7.6: Study of the influence of the neighborhood parameters (β, q) on the biobjective instances for the MEMOTS method ($\square$: $q = 1$, $\times$: $q = 2$, $*$: $q = 3$).

For the parameter β, a value equal to 10 seems reasonable, since there is no more significant improvement after 10. For the parameter q, it appears that to minimize D_1, it is preferable to remove only one item ($q = 1$), except for the 750-3 instance. We can explain that by the fact that removing only one item improves the intensification property of the algorithm. Although, for the 750-3 instance, removing only one item is not sufficient. On the other hand, for the instances with more than 250 items, to minimize D_2, it is better to remove more than one item ($q > 1$). But removing one item in the place of removing more than one item has only a large influence on D_2 for the instances with more than 2 objectives and more than 250 items (500-3 and 750-3).

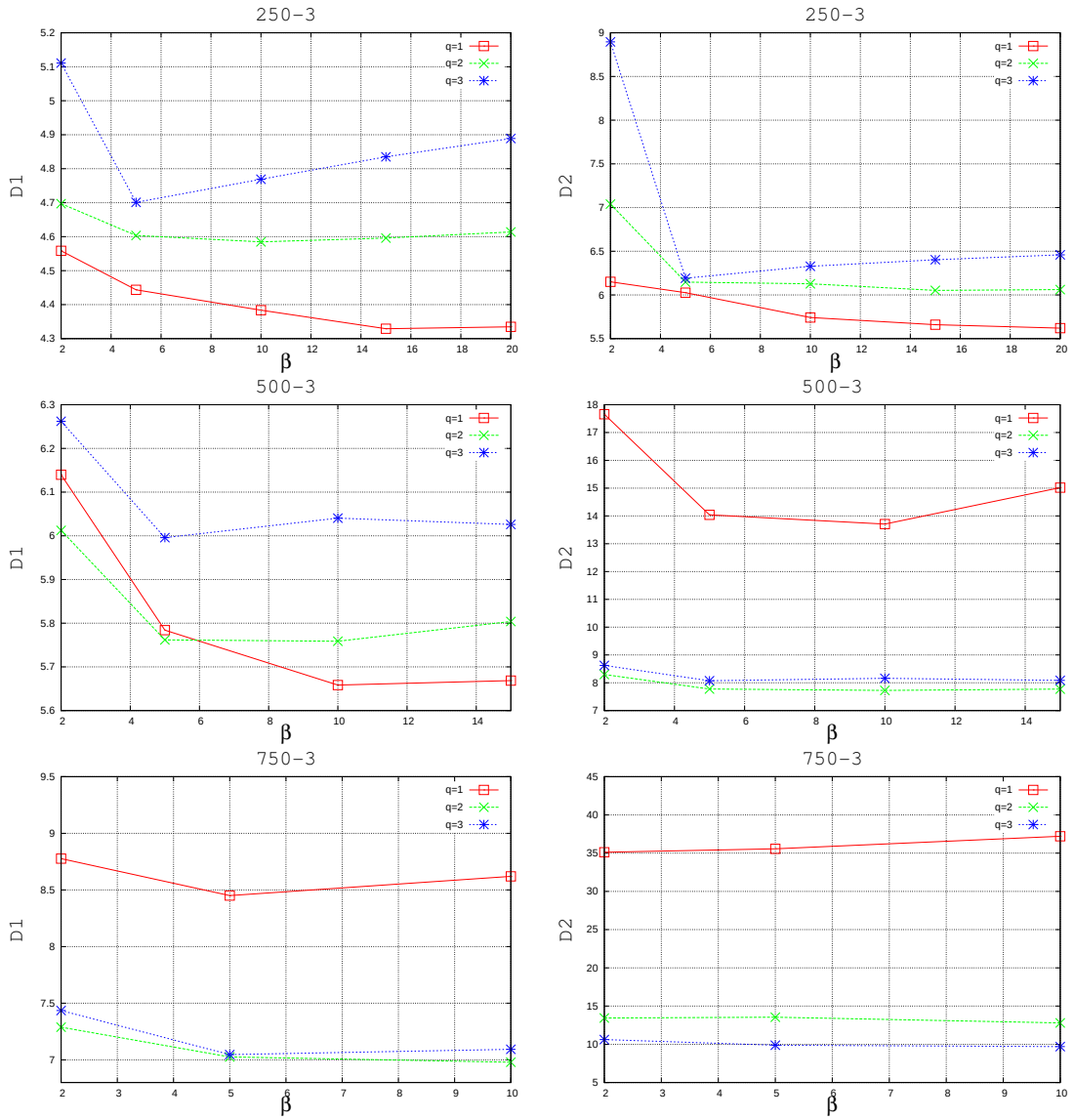

Figure 7.7: Study of the influence of the neighborhood parameters (β, q) on the three-objective instances for the MEMOTS method ($\square$: $q = 1$, $\times$: $q = 2$, $*$: $q = 3$).

Influence of the density threshold

We study in this section the influence of the mean density threshold from which the number of divisions of the hypergrid is increased. We can see in Figure 7.8 the influence of this threshold on D_1 and D_2 for the 750-2 and 250-3 instances.

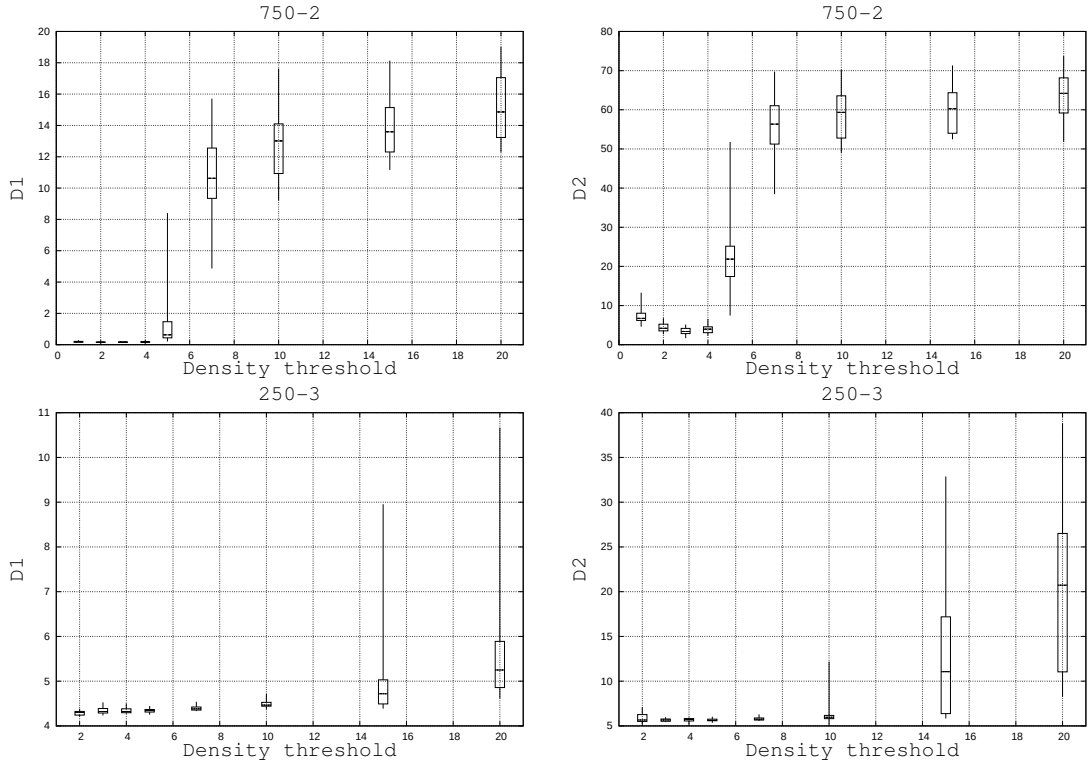

Figure 7.8: Study of the influence of the density threshold on MEMOTS.

We note that a value equal to 3 seems the best value for both instances and using a value larger than 4 for the biobjective instance and larger than 10 for the three-objective instance strongly degrades the performances. We also studied the influence of the threshold on the other instances and we obtained the same conclusion.

Therefore, as mentioned in section 6.2.2 of chapter 6, it is quite easy to set this parameter, since a value around 3 gives good performances for all instances.

Influence of it_{stop}

The influence of the number it_{stop} of iterations without improvement after which PRTS is stopped is represented in Figure 7.9 for the 500-2 and 500-3 instances.

For the 500-2 instance, a value of it_{stop} equal to 1 seems to be an optimal value since this value makes it possible to minimize at the same time D_1 and D_2.

On the other hand, for the 500-3 instance, to take it_{stop} equal to 1 in the place of zero (only one iteration for PRTS) degrades the D_2 indicator, but it_{stop} equal to 1 remains the optimal value for D_1.

We can interpret this difference between the biobjective and three-objective instances by the fact that for the three-objective instances, the number of non-dominated solutions is more important, and for this reason it is preferable to have a diversified research rather than using intensively the local search. By consequence, for the biobjective instances, it_{stop} will be fixed to one and for the three-objective instances, only one iteration of PRTS will be carried out (the tabu list has thus no more impact).

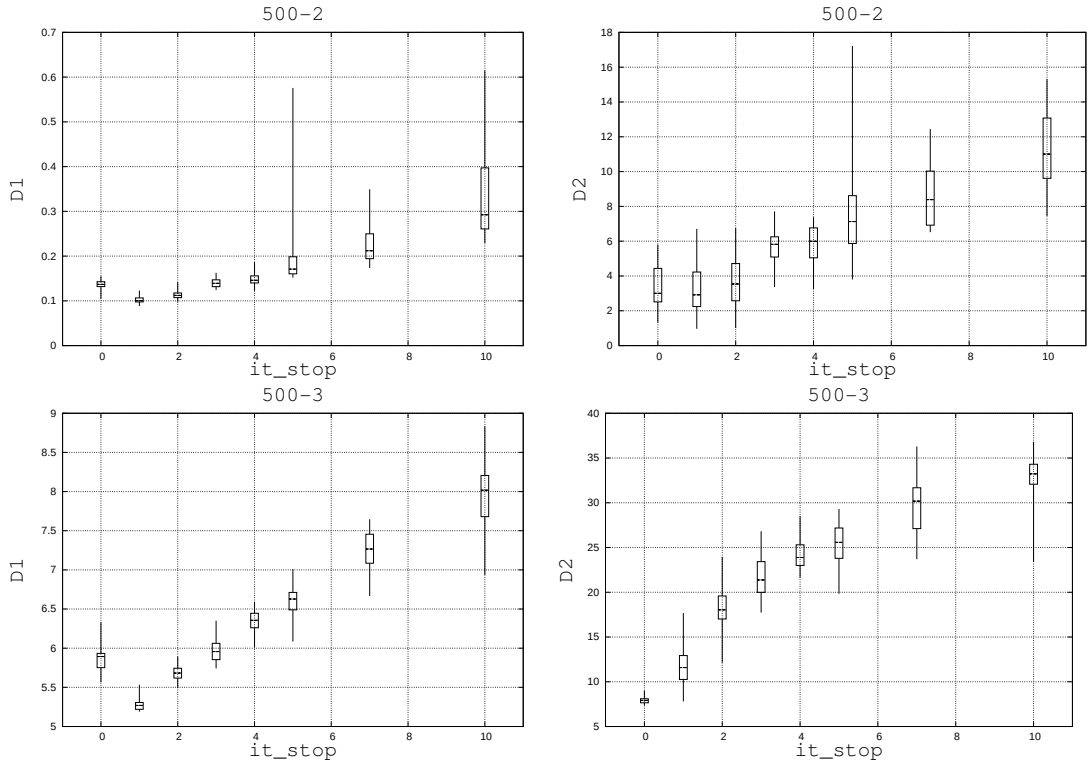

Figure 7.9: Study of the influence of it_{stop} on MEMOTS.

Influence of r

We can see in Figure 7.10 the influence of the parameter r, that is the number of solutions taken into account for the selection of the closest solution to the first parent. For the 750-2 instance, the results show that if r is higher than approximately 30, D_2 is getting worse, which concurs with Hishibuchi [111] on the fact that it is preferable to combine similar parents in multiobjective optimization. For the 250-3 instance, the value of r from which D_2 is getting worse is higher, since the number of potentially efficient solutions obtained by the algorithm is more important.

On the other hand, there is a clear improvement of D_1 if the r value increases, but this improvement stops when the value of r reaches the value from which D_2 increases.

Summary

We give in Table 7.4, good compromise values, between intensification and diversification, based on the preceding experimentations, for the six instances experimented.

Table 7.4: Good compromise values for the parameters of MEMOTS.

Instance	N	β	q	$density$	$step$	t	r	it_{stop}
250-2	7500	10	1	3	5	3	30	1
500-2	10000	10	1	3	5	5	30	1
750-2	12500	10	1	3	5	7	30	1
250-3	10000	10	1	3	5	/	200	0
500-3	12500	5	2	3	5	/	300	0
750-3	15000	5	3	3	5	/	400	0

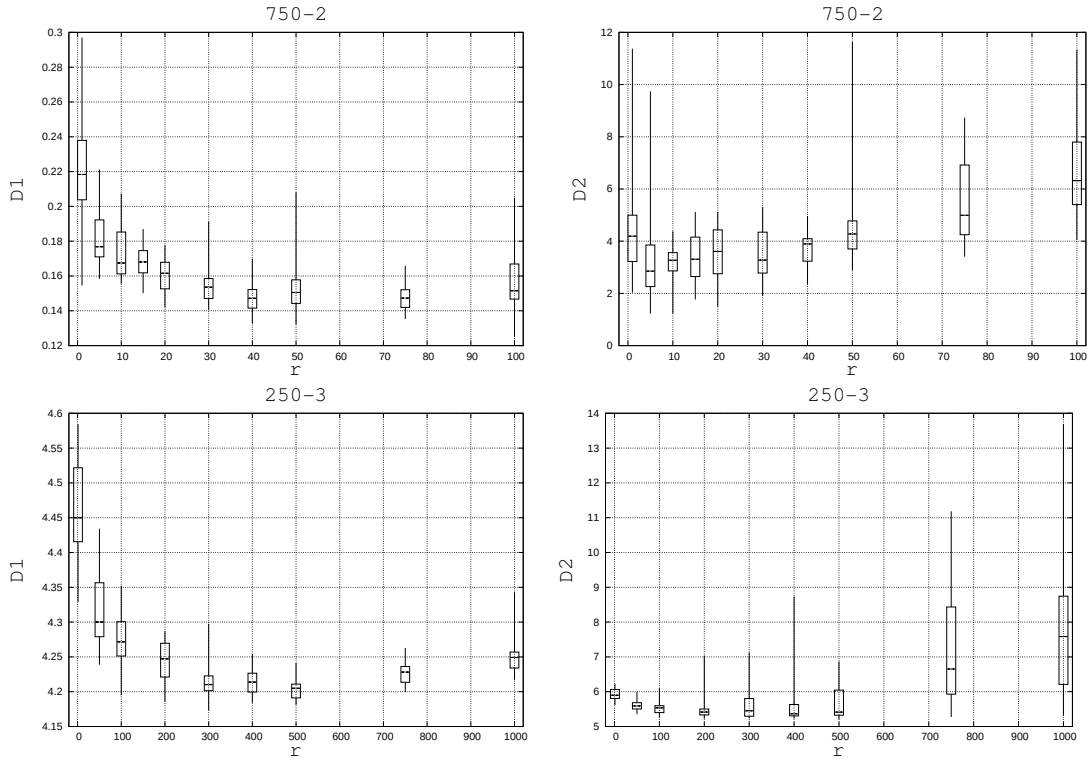

Figure 7.10: Study of the influence of r on MEMOTS.

We now consider these parameters and study two more things: the influence of the initial population and the influence of the number N of iterations.

Influence of the initial population

Two ways of generating the initial population are tested: following P_1 or following P_2 (see section 7.2.3). The P_2 process to generate the initial population takes more time than P_1, whose the running time is insignificant and close to zero second. It is important to see if it is worthwhile or not to take more time to generate the initial population. The results are given in Figure 7.11. The results obtained with P_1 are presented in red, while the results obtained with P_2 are presented in green. We compare the evolution of D_1 and D_2 according to the running time of MEMOTS and the evolution of the running time according to the number of iterations N for both ways of generating the initial population, for the 500-2 and 750-2 instances. For D_1, we see that we obtain slightly better results with P_2: for a same running time, the values of D_1 with P_2 are slightly lower. The variations around D_1 are low, except when P_1 is used with a small number of iterations. For D_2, on the 500-2 instance, the results seem equivalent from a certain running time. For the 750-2 instance, the results obtained with P_2 are better and present less variations. Concerning the running time, for a same number of iterations, the running time with P_2 is higher than P_1; the gap is always equal to more or less the time needed to generate the initial population.

Following these experimentations, we recommend the use of the way P_2 to generate the initial population.

Influence of N

We study here the influence of the number N of iterations. Obviously, the higher N, the better the quality of the results is. But it is interesting to see from which value of N the quality of the

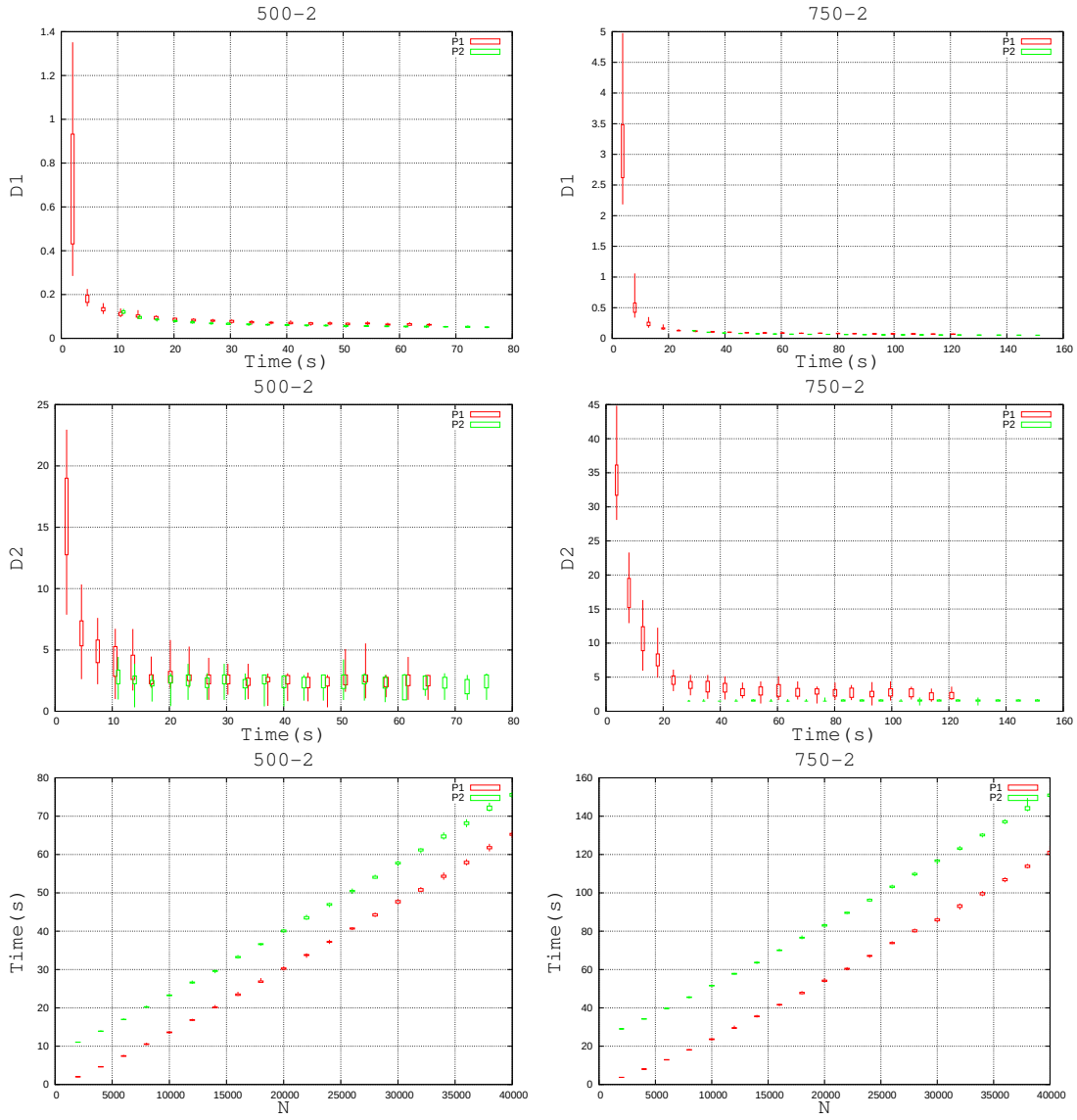

Figure 7.11: Study of the influence of the initial population on MEMOTS.

results is leveling off. The results have been obtained with the parameters of Table 7.4. The initial population has been generated following P_2.

We show the results in Figure 7.12 for the 750-2 and 750-3 instances. As the running time strongly depends on N, we also show its evolution (but it does not include the time needed to generate the initial population, that does not depend on N).

For the 750-2 instance, we remark that from N equal to about 12500, there is no more strong improvement in D_1. The best values for D_2 are already reached with a small number of iterations. The variations around D_1 are small in comparison with the variations around D_2. The running time evolves more or less linearly according to the number of iterations N. The number $|PE|$ of potentially efficient solutions found evolves logarithmically, that is, it is more and more difficult to generate new potentially efficient solutions.

For the 750-3 instance, the main difference compared to the biobjective instance is that the number of potentially efficient solutions is very high (until 40000) and the running time, contrary to the biobjective instances, is not anymore evolving linearly. It takes indeed more time to update a high size potentially efficient set (obtained with a high number of iterations)

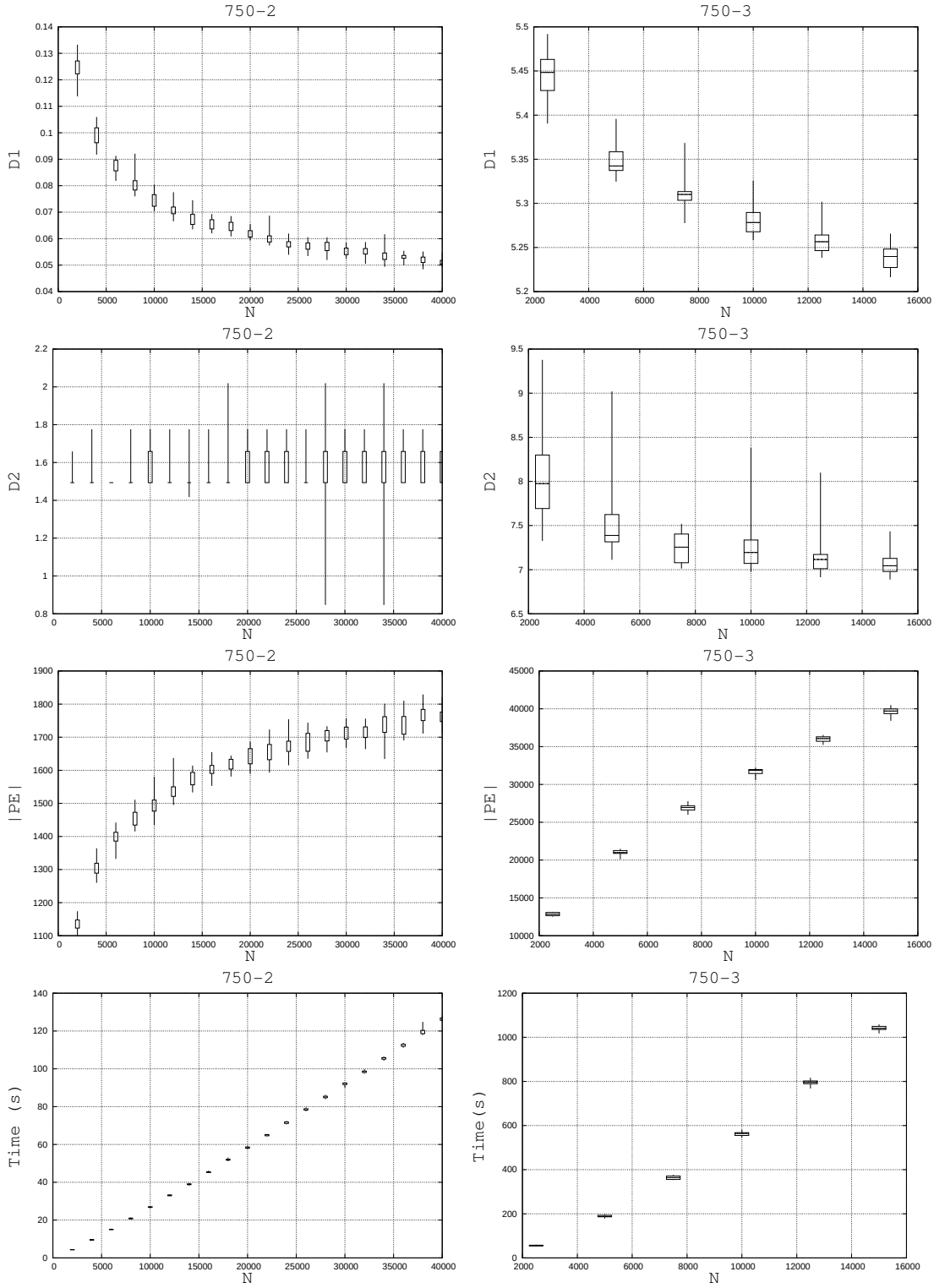

Figure 7.12: Study of the influence of N on MEMOTS for the 750-2 and 750-3 instances (comparison between P_1 and P_2).

than one of low size, especially when three objectives are considered. Also, the values of D_2 decreases according to N, which was not the case in the biobjective case.

7.5.3 Study of the influence of the parameters of 2PPLS

As shown in Figure 7.13, 2PPLS presents different alternatives: four for the initial population (whose the way P_3 requires the number S of weight sets) and three for the neighborhood. Moreover, if the neighborhood $\mathcal{N}_2$ is considered, the size L of the list used to create the small instance of the MOMKP has to be set. Then, if the simplified version of MEMOTS is employed to solve this small instance, the way to generate the initial population of MEMOTS has to be defined, the number N of iterations has to be fixed as well as the parameters of PRTS used in MEMOTS, that is it_{stop} (maximal number of iterations of PRTS without improvement), the tabu tenure t and the neighborhood parameters (β, q). But in this case, we only study the influence of N. The parameters it_{stop} and t have respectively been fixed to 1 and 3. For the neighborhood, β is equal to 100 and q to 1. For the initial population of MEMOTS, we use the alternative P_3 (see section 7.3.1) with S equal to 50.

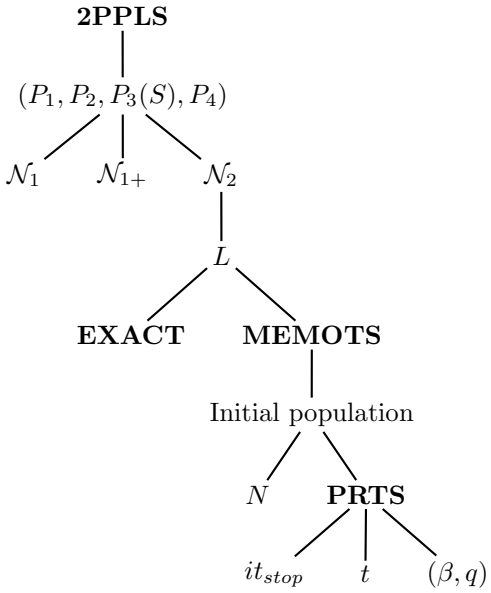

Figure 7.13: Parameters of 2PPLS.

We only study the influence of the parameters of 2PPLS for the biobjective instances, mainly because 2PPLS is better suited for solving biobjective problems (see section 6.3.1 in chapter 6, and that we will also be shown latter in section 7.8.2).

We first proceed a comparison between the neighborhoods. The different alternatives for generating the initial population will be then studied.

Neighborhood

We first compare the neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_{1+}$ that do not require any parameters. Concerning the initial population of 2PPLS, the option P_2 (see section 7.3.1) has been selected to run the experiments.

In this subsection, the running time indicated corresponds only to the second phase, as the running time of the first phase is the same for each neighborhood experimented.

In Tables 7.5 and 7.6 we can see the comparison of the indicators obtained by the neighborhoods. We also add the results of MEMOTS (with the parameters of Table 7.4, without taking into account the running time of the generation of the initial population following P_2) to

have a better view of the quality of the results. We use five different indicators to measure the quality of the approximations obtained: the $\mathcal{H}$, $I_{\epsilon 1}$, R, D_1 and D_2 indicators (see section 3.5.1 in chapter 3). We also add as additional information the number $|PE|$ of potentially efficient solutions generated and the running time in seconds.

We first remark that the running time with $\mathcal{N}_{1+}$ is much higher than the running time with $\mathcal{N}_1$. The results obtained with $\mathcal{N}_{1+}$ are better, but only slightly. Comparing to MEMOTS, for 250-2, MEMOTS is better on all the indicators but $I_{\epsilon 1}$. For 500-2, MEMOTS is better on $\mathcal{H}$, D_1, D_2, $P_{Y_{SN}}$ and $P_{Y_\triangle}$. For 750-2, 2PPLS-$\mathcal{N}_{1+}$ is better on all indicators, but $\mathcal{H}$ and $P_{Y_{SN}}$; 2PPLS-$\mathcal{N}_{1+}$ seems thus slightly better than MEMOTS, but its running time is much higher. On the other hand, for 750-2, 2PPLS-$\mathcal{N}_1$ has a running time lower than MEMOTS and gives better results for $I_{\epsilon 1}$, R, D_1, D_2 and $P_{Y_\triangle}$. The neighborhood $\mathcal{N}_1$ by its simplicity and its relatively good results appears thus as a more interesting alternative than $\mathcal{N}_{1+}$. It seems that the addition of running time in $\mathcal{N}_{1+}$ is not worthwhile in comparison with the slight improvement of the results. Thus, the neighborhood $\mathcal{N}_{1+}$ does not seem to be an efficient neighborhood, in term of ratio quality/computational time.

Table 7.5: Comparison of 2PPLS with $\mathcal{N}_1$ or $\mathcal{N}_{1+}$, based on the indicators (1).

Instance	Algorithm	$\mathcal{H}(10^7)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
250-2	2PPLS-$\mathcal{N}_1$	9.8631	1.001160	246.715037	0.128	2.961	344.45	**0.68**
	2PPLS-$\mathcal{N}_{1+}$	9.8648	**1.001116**	246.518029	0.093	2.026	377.95	3.99
	MEMOTS	**9.8699**	1.001122	**246.469702**	**0.072**	**0.822**	394.15	4.05
500-2	2PPLS-$\mathcal{N}_1$	40.7708	1.000551	431.899185	0.084	2.331	793.75	**5.47**
	2PPLS-$\mathcal{N}_{1+}$	40.7714	**1.000503**	**431.746906**	0.074	2.370	879.15	45.76
	MEMOTS	**40.7817**	1.000523	431.850981	**0.073**	**2.235**	846.85	14.65
750-2	2PPLS-$\mathcal{N}_1$	89.3468	1.000407	743.216602	0.065	1.494	1745.20	**30.17**
	2PPLS-$\mathcal{N}_{1+}$	89.3479	**1.000330**	**743.010747**	**0.056**	**1.494**	1883.65	328.01
	MEMOTS	**89.3498**	1.000410	743.496760	0.070	1.579	1538.55	34.78

Table 7.6: Comparison of 2PPLS with $\mathcal{N}_1$ or $\mathcal{N}_{1+}$, based on the indicators (2).

Instance	Algorithm	P_{Y_N}(%)	$P_{Y_{SN}}$(%)	$P_{Y_{NN}}$(%)	$P_{Y_\triangle}$(%)
250-2	2PPLS-$\mathcal{N}_1$	15.48	27.75	14.55	83.96
	2PPLS-$\mathcal{N}_{1+}$	20.36	38.00	19.02	88.75
	MEMOTS	**25.15**	**52.75**	**23.06**	**90.42**
500-2	2PPLS-$\mathcal{N}_1$	4.57	7.21	4.42	81.91
	2PPLS-$\mathcal{N}_{1+}$	**6.79**	14.09	**6.37**	84.36
	MEMOTS	6.31	**14.68**	5.83	**84.90**
750-2	2PPLS-$\mathcal{N}_1$	0.96	3.03	0.88	82.82
	2PPLS-$\mathcal{N}_{1+}$	**1.75**	2.80	**1.71**	**85.80**
	MEMOTS	1.24	**4.22**	1.13	82.73

For $\mathcal{N}_2$, we show in Figure 7.14 the evolution of the indicators D_1, D_2, P_{Y_N} and the running time according to L, for the EXACT subroutine to solve the small instances of the MOMKP. We vary the value of L from 4 to 8. We see that there are strong improvements of the indicators when L is increased, except for D_2. On the other hand, the running times evolve exponentially according to L, as expected since the EXACT subroutine is a simple enumeration. For both instances, using a value of L superior to 8 would give unreasonable running times.

In Figure 7.15, we show the evolution of the indicators D_1, P_{Y_N} and the running time according to L, in case of the use of the MEMOTS subroutine to solve the small instances of the MOMKP. We vary the values of L from 4 to 20. We use three different numbers of iterations: $N = 100$ (in red), $N = 200$ (in green) and $N = 400$ (in blue). We see that for small values of L,

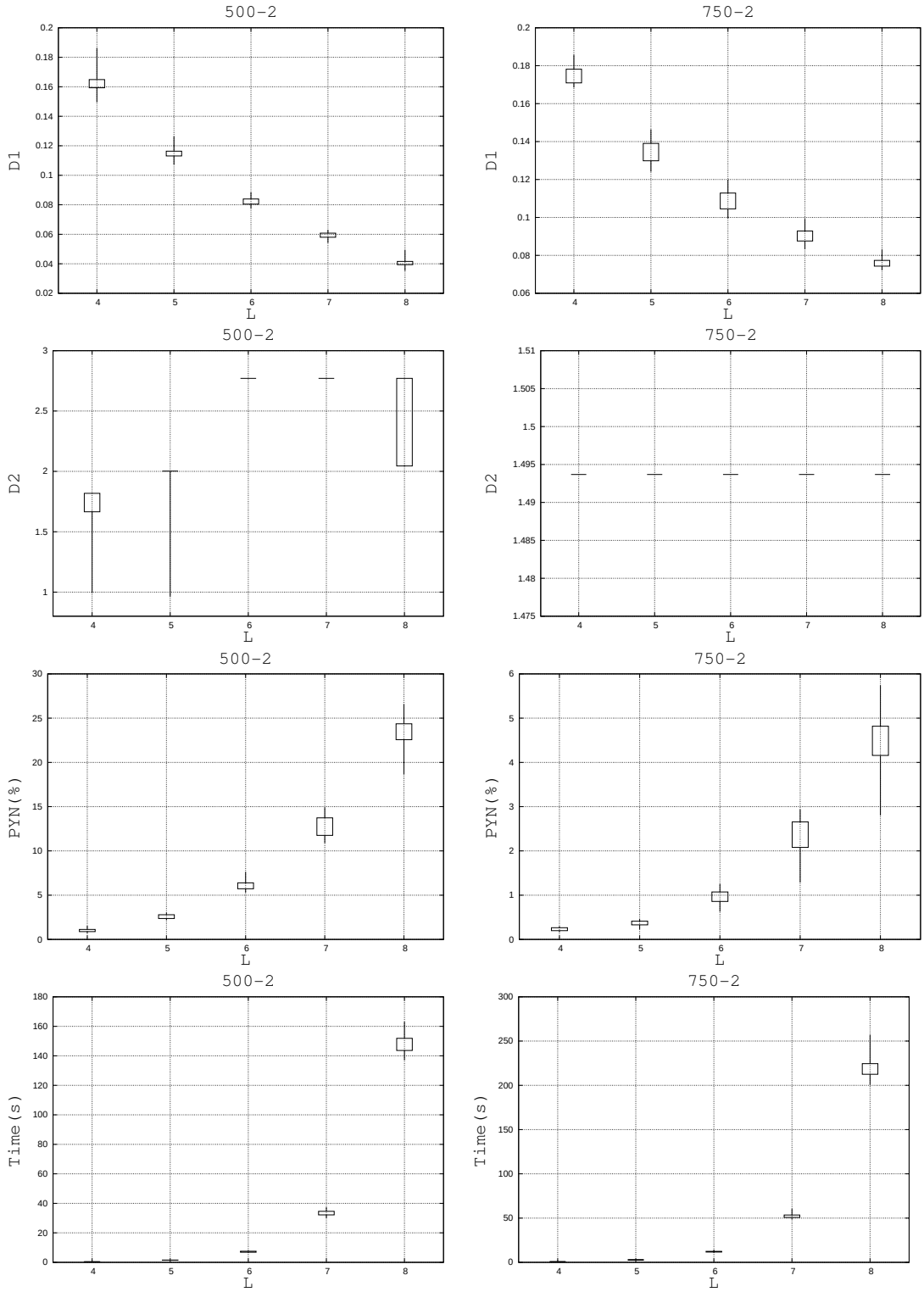

Figure 7.14: Study of the influence of the value of L for the EXACT subroutine to solve the small MOMKP instances (neighborhood $\mathcal{N}_2$).

the indicators D_1 and P_{Y_N} are more or less equal no matter what the number of iterations is. From L equal to 10, it is clear that we obtain better results if N is higher. On the other hand, the running time is bigger when N is higher, but still evolves more or less linearly according to L. An interesting behavior is pointed by the figure showing the evolution of P_{Y_N} according to L for the 500-2 instance. From L equal to about 16 and for a number of iterations N equal to 100 or 200, there is a deterioration of P_{Y_N} while the running time is increasing. It means that the number of iterations is not high enough to solve the small instances of the MOMKP, and that therefore the quality of the approximations obtained for the small instances is not good enough to improve P_{Y_N}. Fixing good values for L and N seems thus not easy since these two values have to be increased at the same time if we want to improve the qualities of the results.

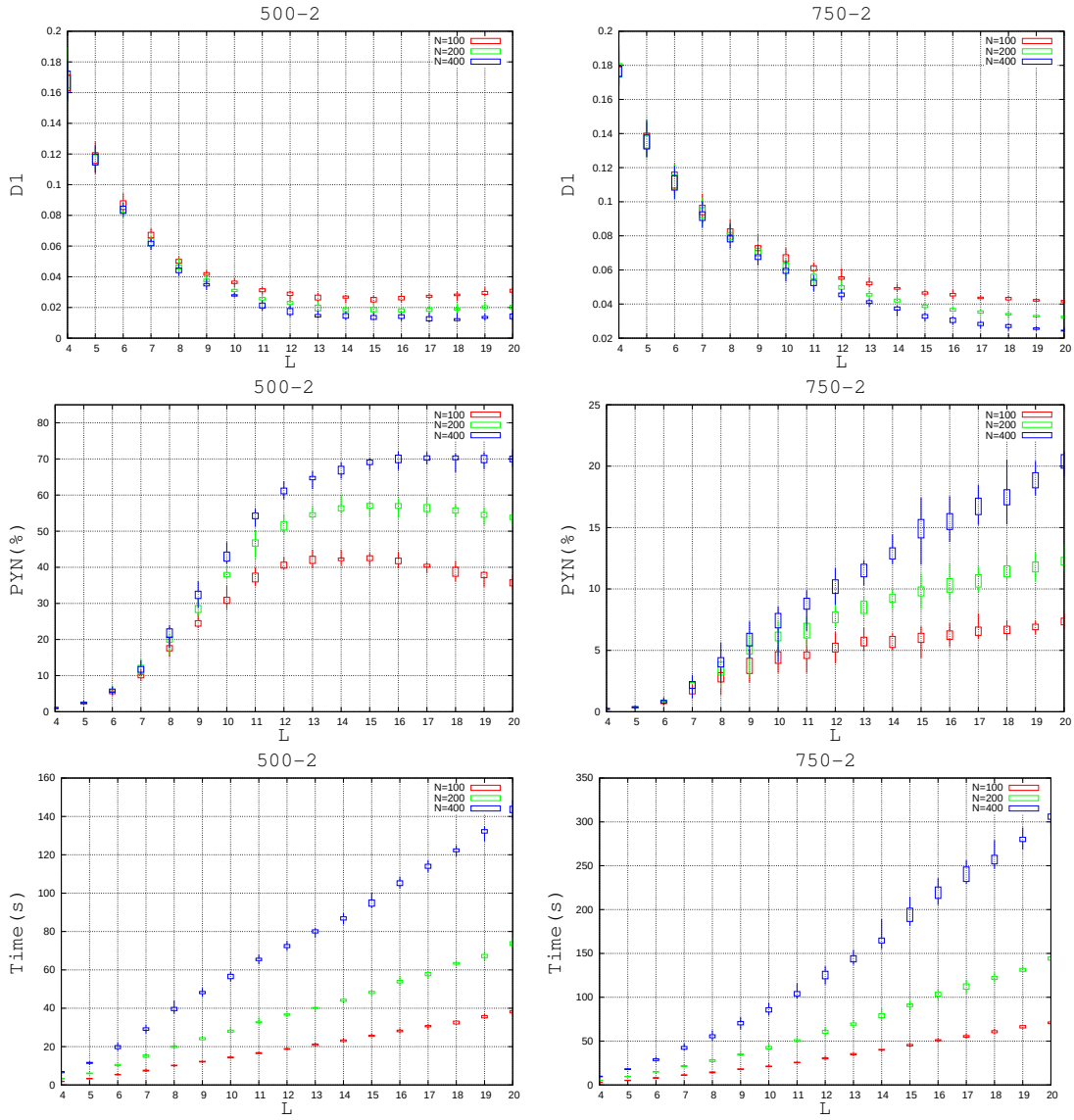

Figure 7.15: Study of the influence of the value of L for the MEMOTS subroutine to solve the small MOMKP instances (neighborhood $\mathcal{N}_2$).

Now, it is interesting to see, for a fixed running time, what is the best to do: using the EXACT or MEMOTS subroutine, with which value of L, and for MEMOTS, with how many iterations.

In order to answer this question, we have represented in Figure 7.16, the evolution of D_1

and P_{Y_N} according to the running time (the running time is controlled by the value of L: from 4 to 8 for the EXACT subroutine and from 4 to 20 for the MEMOTS subroutine). We see that for only very small running times, it is better to use the EXACT subroutine. As soon as we give more running times, it is better to use the MEMOTS subroutine. In this case, the good combination between L and N has to be determined according to the running time given. The higher the running time allowed, the higher the number of iterations should be.

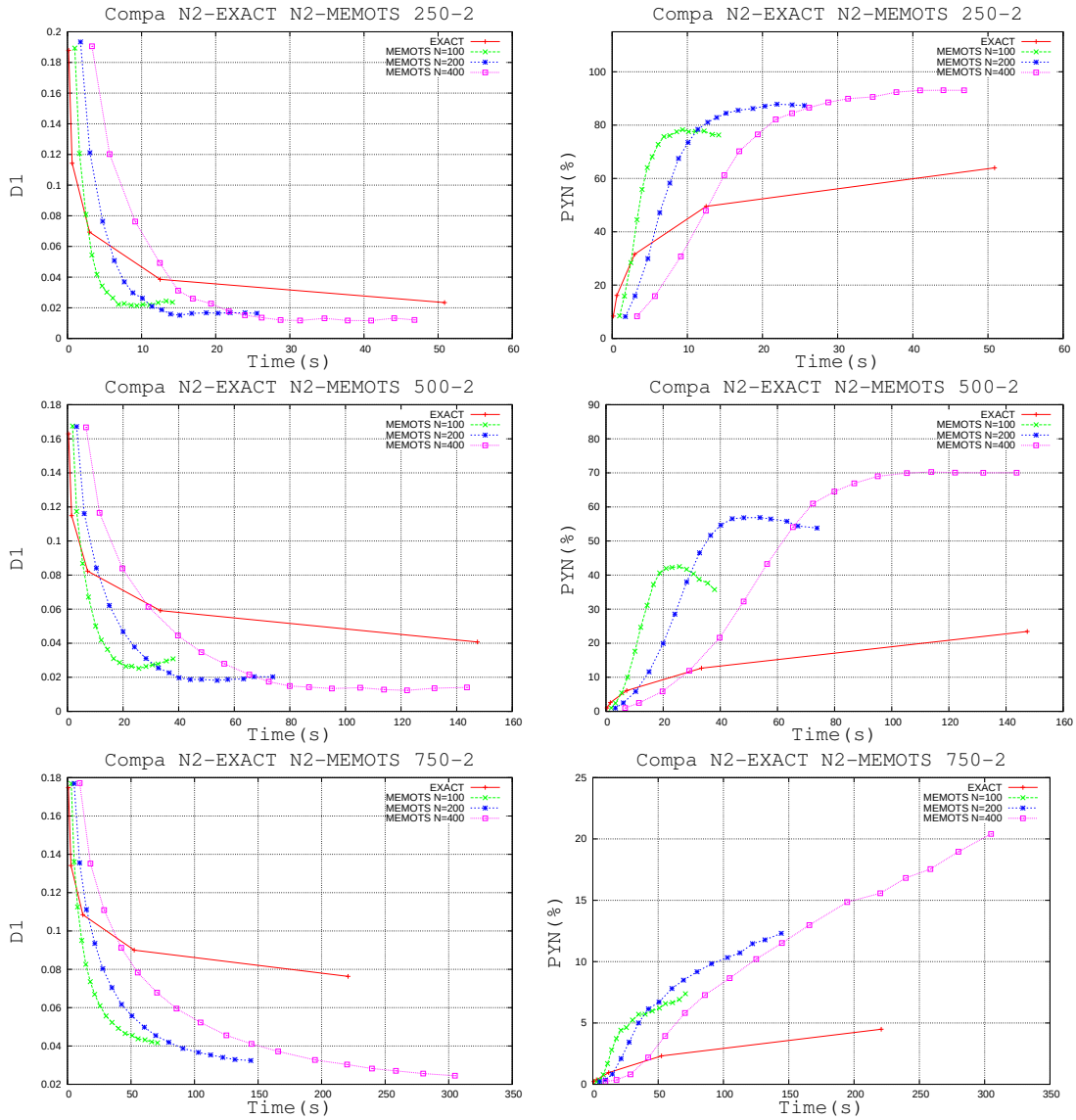

Figure 7.16: Comparison between the EXACT and MEMOTS subroutines to solve the small MOMKP instances according to the running time (neighborhood $\mathcal{N}_2$).

Initial population

We analyze here the influence of the initial population on the results of 2PPLS. We first assess the quality of the initial population. We consider the four ways to generate the initial population presented in section 7.3.1. For the way P_3, we present the results with S=50 and S=1000. The results are presented in Tables 7.7 and 7.8. We see that the way P_1 is very time-consuming. However, with this method, we generate 100% of the supported non-dominated points for the

250-2 and 750-2 instances. For the 500-2 instance, only 98.7% of the supported non-dominated points are generated given that there exist non-extreme supported non-dominated points for this instance. We see that the way P_3 with $S = 1000$ allows to generate more potentially non-dominated points than P_1, since with P_3, potentially non-supported non-dominated points are generated. Thereof, the distance D_1 of P_3 with $S = 1000$ is lower than with P_1. The results of P_3 with S=1000 are slightly better than with P_2, but the running time of P_3 with S=1000 is a bit higher. The way P_4 and the way P_3 with S=50 are very fast, but the results are poor in comparison with the other alternatives.

Table 7.7: Comparison of the different ways to generate the initial population of 2PPLS, based on the indicators (1).

Instance	Init Pop	$\mathcal{H}(10^7)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
250-2	P_1	**9.8656**	**1.005282**	**251.143630**	0.611	**2.299**	40.00	2252.44
	P_2	9.8530	1.010459	256.780552	0.720	3.876	42.50	1.50
	P_3 ($S = 50$)	9.8546	1.011867	257.986984	0.815	4.033	37.00	0.16
	P_3 ($S = 1000$)	9.8582	1.010082	254.846633	**0.582**	3.259	51.00	3.09
	P_4	9.8456	1.011916	262.027902	1.206	4.638	26.00	0.13
500-2	P_1	**40.7838**	1.003457	**437.164134**	0.459	**1.544**	76.00	22824.50
	P_2	40.7362	1.005027	442.242904	0.575	2.486	73.75	8.60
	P_3 ($S = 50$)	40.7289	1.004888	445.641000	0.784	2.626	46.00	0.63
	P_3 ($S = 1000$)	40.7441	**1.002725**	438.147294	**0.380**	2.098	116.00	12.66
	P_4	40.7224	1.008061	449.138624	0.989	3.700	37.00	0.68
750-2	P_1	**89.3446**	**1.002361**	**747.982769**	0.365	1.551	109.00	40451.67
	P_2	89.3059	1.005047	756.844196	0.569	2.525	109.90	24.67
	P_3 ($S = 50$)	89.2724	1.007196	768.141067	1.031	3.769	48.00	1.39
	P_3 ($S = 1000$)	89.3221	1.002845	751.086985	**0.349**	**1.494**	182.00	27.56
	P_4	89.2752	1.007397	766.495756	1.020	3.351	49.00	1.80

Table 7.8: Comparison of the different ways to generate the initial population of 2PPLS, based on the indicators (2).

Instance	Init Pop	$P_{Y_N}(\%)$	$P_{Y_{SN}}(\%)$	$P_{Y_{NN}}(\%)$	$P_{Y_\triangle}(\%)$
250-2	P_1	**7.04**	**100.00**	0.00	**100.00**
	P_2	0.75	5.00	0.43	56.10
	P_3 ($S = 50$)	1.06	10.00	0.38	56.76
	P_3 ($S = 1000$)	1.58	12.50	**0.76**	68.63
	P_4	0.53	2.50	0.38	61.54
500-2	P_1	**5.37**	**98.70**	0.00	**100.00**
	P_2	0.08	0.65	0.05	45.72
	P_3 ($S = 50$)	0.14	2.60	0.00	45.65
	P_3 ($S = 1000$)	0.35	5.19	**0.07**	53.45
	P_4	0.00	0.00	0.00	35.14
750-2	P_1	**3.60**	**100.00**	0.00	**100.00**
	P_2	0.08	0.78	**0.05**	46.51
	P_3 ($S = 50$)	0.00	0.00	0.00	37.50
	P_3 ($S = 1000$)	0.07	0.92	0.03	54.40
	P_4	0.00	0.00	0.00	46.94

We now compare the results of the different ways of generating the initial population, plus the application of PLS. We first show the results with the neighborhood $\mathcal{N}_2$, and the MEMOTS subroutine with $L = 12$ and $N = 100$. The results are showed in Tables 7.9 and 7.10. The

Table 7.9: Comparison of P_i + PLS-$\mathcal{N}_2$-MEM, based on the indicators (1).

Instance	Algorithm	$\mathcal{H}(10^7)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time PLS(s)
250-2	P_1+PLS-$\mathcal{N}_2$-MEM	**9.8710**	1.000500	**245.686422**	**0.017**	**0.698**	505.70	6.82
	P_2+PLS-$\mathcal{N}_2$-MEM	9.8691	**1.000480**	245.691015	0.022	2.458	505.85	6.88
	$P_3(S = 500)$+PLS-$\mathcal{N}_2$-MEM	9.8686	1.000487	245.700740	0.025	2.658	506.40	**6.64**
	P_4+PLS-$\mathcal{N}_2$-MEM	9.8691	1.000484	245.691155	0.023	2.548	506.15	7.93
500-2	P_1+PLS-$\mathcal{N}_2$-MEM	**40.7937**	1.000267	**430.884486**	**0.025**	**0.408**	1083.75	**17.68**
	P_2+PLS-$\mathcal{N}_2$-MEM	40.7867	1.000285	430.905228	0.029	2.778	1113.40	18.77
	$P_3(S = 500)$+PLS-$\mathcal{N}_2$-MEM	40.7875	1.000288	430.908349	0.028	2.640	1116.05	18.34
	P_4+PLS-$\mathcal{N}_2$-MEM	40.7869	**1.000261**	430.890851	0.028	2.789	1116.00	21.44
750-2	P_1+PLS-$\mathcal{N}_2$-MEM	**89.3608**	**1.000326**	**742.729416**	**0.049**	**0.612**	1623.55	**26.58**
	P_2+PLS-$\mathcal{N}_2$-MEM	89.3491	1.000366	743.044997	0.056	1.546	1776.30	30.38
	$P_3(S = 500)$+PLS-$\mathcal{N}_2$-MEM	89.3510	1.000377	742.972838	0.054	1.524	1765.80	28.25
	P_4+PLS-$\mathcal{N}_2$-MEM	89.3510	1.000522	743.007395	0.056	1.546	1773.95	37.50

Table 7.10: Comparison of P_i + PLS-$\mathcal{N}_2$-MEM, based on the indicators (2).

Instance	Algorithm	$P_{Y_N}(\%)$	$P_{Y_{SN}}(\%)$	$P_{Y_{NN}}(\%)$	$P_{Y_\triangle}(\%)$
250-2	P_1+PLS-$\mathcal{N}_2$-MEM	74.93	**100.00**	73.03	**100.00**
	P_2+PLS-$\mathcal{N}_2$-MEM	**75.70**	88.88	**74.70**	99.61
	$P_3(S = 500)$+PLS-$\mathcal{N}_2$-MEM	75.18	87.50	74.25	99.47
	P_4+PLS-$\mathcal{N}_2$-MEM	75.48	89.38	74.43	99.66
500-2	P_1+PLS-$\mathcal{N}_2$-MEM	38.13	**99.74**	34.59	**99.95**
	P_2+PLS-$\mathcal{N}_2$-MEM	40.61	74.09	38.69	98.23
	$P_3(S = 500)$+PLS-$\mathcal{N}_2$-MEM	39.83	71.56	38.00	98.03
	P_4+PLS-$\mathcal{N}_2$-MEM	**41.00**	73.70	**39.12**	98.17
750-2	P_1+PLS-$\mathcal{N}_2$-MEM	**8.01**	**100.00**	4.58	**100.00**
	P_2+PLS-$\mathcal{N}_2$-MEM	5.24	15.05	4.87	88.38
	$P_3(S = 500)$+PLS-$\mathcal{N}_2$-MEM	5.59	15.96	**5.20**	88.90
	P_4+PLS-$\mathcal{N}_2$-MEM	4.71	11.01	4.47	86.99

running time presented corresponds only to the running time of the second phase. For P_3, we use S equal to 500. We see that the results are very similar for the four different ways of generating the initial population. The best results are however obtained with the way P_1, but the results obtained with the other ways are very close, except for the D_2 and $P_{Y_{SN}}$ indicators. The running times with P_4 are a little bit higher.

Therefore, the quality of the initial population does not seem to be a crucial point when PLS is used with the neighborhood $\mathcal{N}_2$ and the MEMOTS subroutine. The neighborhood is sufficiently good and PLS can be started from a second-rate initial population.

We present in Tables 7.11 and 7.12 the results obtained with the different populations by application of PLS with the neighborhood $\mathcal{N}_1$. We see that the results with the different populations are also very close, with the best results obtained with P_1. On the other hand, the running time of PLS with P_4 is higher, especially for the 750-2 instance.

Table 7.11: Comparison of P_i + PLS-$\mathcal{N}_1$, based on the indicators (1).

Instance	Algorithm	$\mathcal{H}(10^7)$	$I_{\epsilon 1}$	R	D_1	D_2	$\lvert PE\rvert$	Time(s)
250-2	P_1+PLS-$\mathcal{N}_1$	**9.8699**	1.001126	**246.540945**	**0.110**	**1.356**	300.00	**0.49**
	P_2+PLS-$\mathcal{N}_1$	9.8631	1.001160	246.715037	0.128	2.961	344.45	0.68
	$P_3(S=500)$+PLS-$\mathcal{N}_1$	9.8658	**1.001098**	246.648914	0.123	2.838	332.00	0.68
	P_4+PLS-$\mathcal{N}_1$	9.8630	1.001419	246.837124	0.137	2.961	338.00	0.81
500-2	P_1+PLS-$\mathcal{N}_1$	**40.7919**	**1.000466**	**431.732199**	0.083	**0.643**	707.00	**4.34**
	P_2+PLS-$\mathcal{N}_1$	40.7708	1.000551	431.899185	0.084	2.331	793.75	5.47
	$P_3(S=500)$+PLS-$\mathcal{N}_1$	40.7718	1.000493	431.769366	**0.078**	1.162	782.00	5.41
	P_4+PLS-$\mathcal{N}_1$	40.7695	1.000947	432.017433	0.092	3.500	799.00	6.46
750-2	P_1+PLS-$\mathcal{N}_1$	**89.3601**	**1.000300**	**742.719423**	**0.057**	**0.910**	1452.00	**21.18**
	P_2+PLS-$\mathcal{N}_1$	89.3468	1.000407	743.216602	0.065	1.494	1745.20	30.17
	$P_3(S=500)$+PLS-$\mathcal{N}_1$	89.3473	1.000333	743.048500	0.059	1.494	1753.00	31.74
	P_4+PLS-$\mathcal{N}_1$	89.3456	1.000532	743.334475	0.074	1.538	1801.00	35.41

Table 7.12: Comparison of P_i + PLS-$\mathcal{N}_1$, based on the indicators (2).

Instance	Algorithm	$P_{Y_N}(\%)$	$P_{Y_{SN}}(\%)$	$P_{Y_{NN}}(\%)$	$P_{Y_\triangle}(\%)$
250-2	P_1+PLS-$\mathcal{N}_1$	16.37	**100.00**	10.04	**100.00**
	P_2+PLS-$\mathcal{N}_1$	15.48	27.75	14.55	83.96
	$P_3(S=500)$+PLS-$\mathcal{N}_1$	**16.55**	37.50	**14.96**	87.05
	P_4+PLS-$\mathcal{N}_1$	13.56	25.00	12.69	82.25
500-2	P_1+PLS-$\mathcal{N}_1$	**8.55**	**98.70**	3.36	**99.72**
	P_3+PLS-$\mathcal{N}_1$	4.57	7.21	4.42	81.91
	$P_3(S=500)$+PLS-$\mathcal{N}_1$	5.44	10.39	**5.15**	83.76
	P_4+PLS-$\mathcal{N}_1$	2.68	2.60	2.69	78.85
750-2	P_1+PLS-$\mathcal{N}_1$	**4.69**	**100.00**	1.13	**100.00**
	P_2+PLS-$\mathcal{N}_1$	0.96	3.03	0.88	82.82
	$P_3(S=500)$+PLS-$\mathcal{N}_1$	1.32	4.59	**1.20**	84.54
	P_4+PLS-$\mathcal{N}_1$	0.56	0.00	0.58	80.51

In Figure 7.17 we show the evolutions of D_1 and the running time according to S for the P_3+PLS-$\mathcal{N}_1$ method. For the running time, we show the evolution of the running time of the generation of P_3 (in red), the running time of PLS (in green) and the total running time (in blue).

We remark that D_1 is decreasing with S, except at the beginning when S moves from 1 to 2. The running time of PLS is also decreasing with S, except when S moves from 1 to 2. The running time of the generation of the initial population increases more or less linearly. The

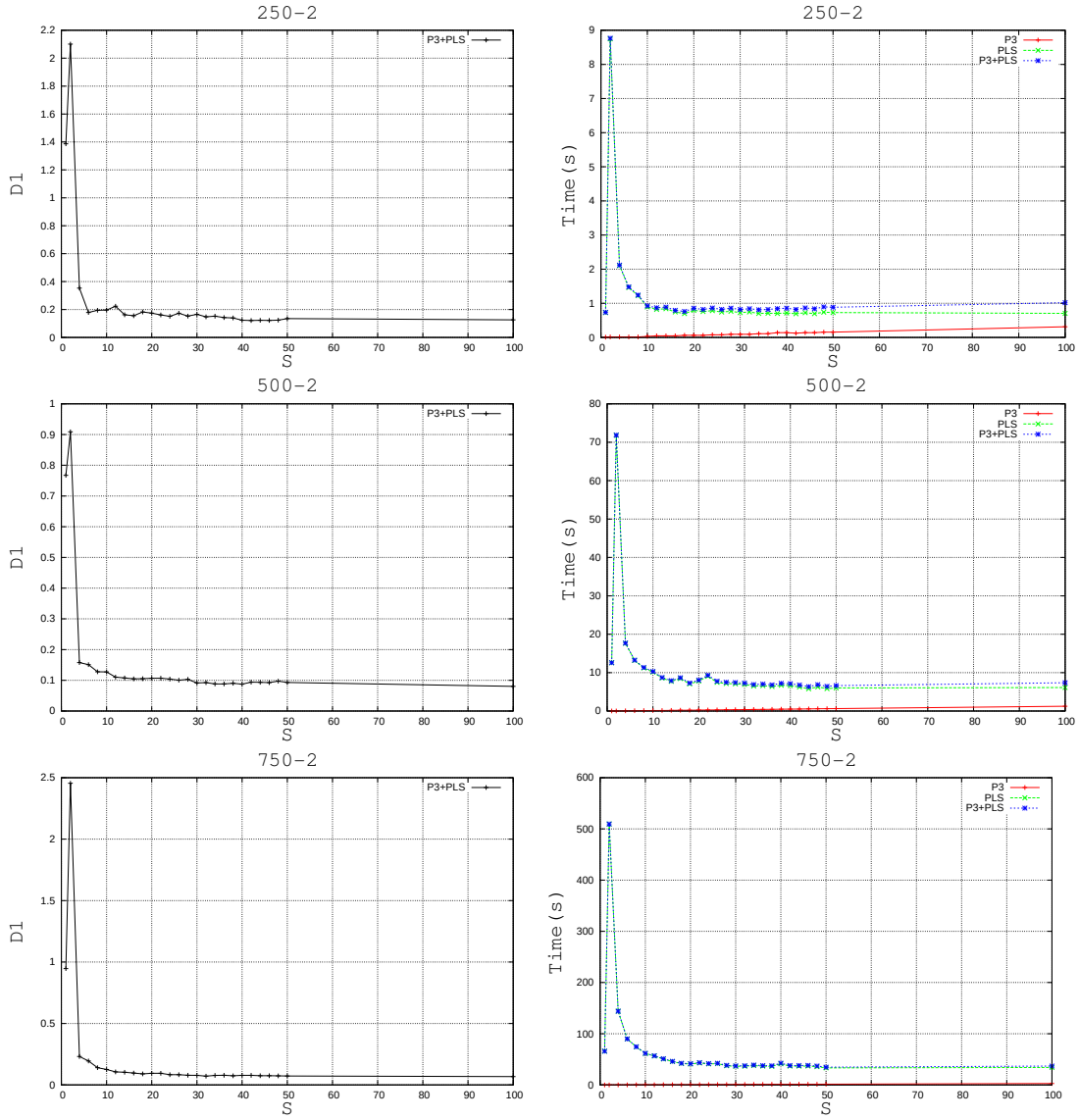

Figure 7.17: Evolution of D_1 and the running time according to S for P_3-PLS-$\mathcal{N}_1$.

total running time presents thus a kind of optimal value, obtained for S equal to about 50. If S becomes higher, the increasing of the performances in term of indicators and running time of the method is not anymore worthwhile in comparison with the increasing of the running time of the first phase. We can explain why there is a degradation of the performances when S moves from 1 to 2: for S=1, only one central initial solution is generated (with the weight set equal to (0.5,0.5)) while with $S = 2$ two initial solutions are generated but there are all located at the extreme region of the Pareto front since there are obtained with the weight sets respectively equal to (1,0) and (0,1). In this case, it is more difficult to obtain good performances in the central part of the Pareto front.

7.6 Values of the parameters: summary

For MEMOTS, the values of the parameters of Table 7.4 give good compromise values, between intensification, diversification and running times. The way P_2 to generate the initial population is recommended. The running time of MEMOTS can be controlled by the number of iterations N. The higher N, the better the approximation is. Values of N after which only small improvements are expected are also given in Table 7.4.

For 2PPLS, the way P_3 with $S = 100$ to generate the initial population and the neighborhood $\mathcal{N}_2$ with the MEMOTS subroutine (with $it_{stop} = 1$, $t = 3$, $\beta = 100$, q to 1 and the alternative P_3 ($S = 50$) to generate the initial population) are advised. The running time of 2PPLS can be controlled by the parameter L, allowing to define the size of the small instance of the MOMKP to solve. However, in order to increase the quality of the approximations according to the running time, it is necessary to increase at the same time the number N of iterations performed in the MEMOTS subroutine. We recommend to vary N linearly according to L in the following way:

$$N = 100 + \frac{75}{4}(L - 4) \tag{7.8}$$

(in this way, for L=4, N=100 and for $L = 20$, N=400).

7.7 Comparison between MEMOTS and 2PPLS on biobjective instances

We realize in this section a comparison between MEMOTS and 2PPLS, for different running times.

For MEMOTS, we use the parameters of Table 7.4, with the way P_2 of generating the initial population. The running time of MEMOTS is controlled with the number of iterations, varying between 2000 and 40000.

For 2PPLS, we use the way P_2 of generating the initial population (in order to use the same initial population as MEMOTS) and the neighborhood $\mathcal{N}_2$ with the MEMOTS subroutine. The running time of 2PPLS is controlled by both L and N. We vary L from 4 to 20, and N linearly evolves according to L following the rule (7.8) defined in the previous section.

The results are presented in Figure 7.18 where the evolutions of D_1 and P_{Y_N} according to the running time (without taking into account the running time of the generation of the initial population, which is the same for both algorithms) are showed.

We see that except with small running times, the results obtained with 2PPLS are better than with MEMOTS. With 2PPLS, we can generate, for the 250-2 instance, about 90% of the non-dominated points, for the 500-2 instance, about 70% and for the 750-2 instance, about 20%, in reasonable running times, which seems a priori remarkable. We will confirm that in the next section, where we compare the results of 2PPLS with state-of-the-art results.

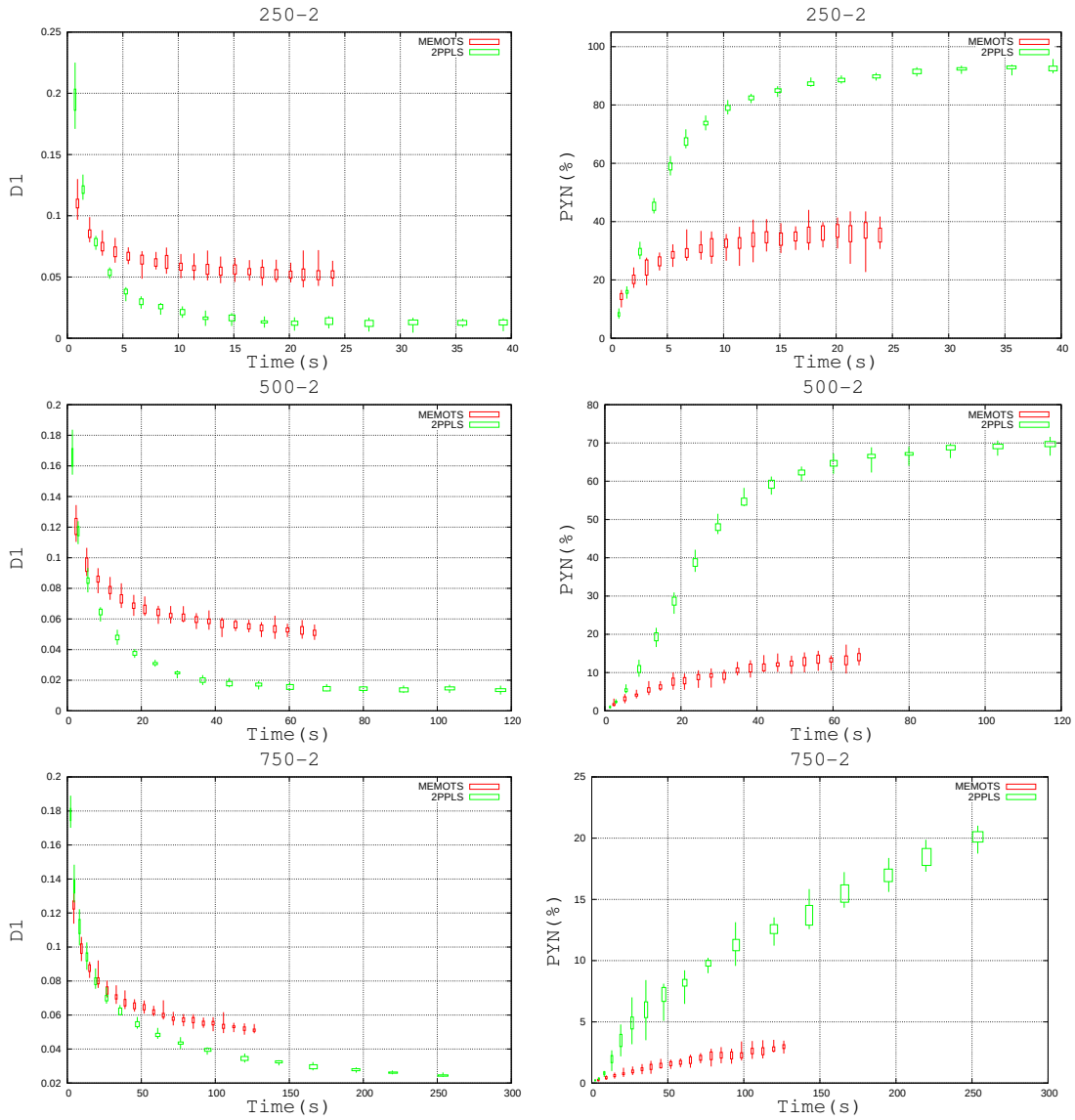

Figure 7.18: Comparison of MEMOTS and 2PPLS: evolution of D_1 and P_{Y_N} according to the running time.

7.8 Comparison with state-of-the-art results

7.8.1 Biobjective instances

Comparison based on the mean of the indicators

We first present the values of the indicators obtained by other algorithms. These results have been obtained either by downloading them from web sites or by asking them personally to the different authors[2].

We have obtained the following results for the 250-2, 500-2, 750-2 instances:

- SPEA [254]: 30 runs.
- SPEA2 [251]: 30 runs, but only for the 750-2 instance.
- MOGLS00 [114]: 20 runs.
- MOGLS04 [114]: 20 runs (different than MOGLS00 since obtained with the library MOMH-Lib++ [113]).
- PMA [120]: 20 runs.
- IMMOGLS [109]: 20 runs.
- MOGTS [15]: 1 run.
- GRASP [238]: 1 run.
- MOTGA [3]: 20 runs.
- PATH-RELINKING [19]: 30 runs.

We see that we have obtained quite a lot of results. It is only a pity that we did not obtain the results of Gomes da Silva *et al.* [36] with their scatter search method.

We present the values of the indicators obtained by these algorithms in Tables 7.13 and 7.14. As we do not know for most of the algorithms the running time, we do not present it and the results are thus only indicative.

We see that the most competitive algorithms are MOGLS04, PMA, IMMOGLS, MOGTS, GRASP and MOTGA. The other algorithms are outdated. Please remark that it is often the results of MOGLS00 that are considered as reference.

In these tables, we also have added the line ALL which gives the indicators obtained by merging the potentially non-dominated points obtained by all the runs of all algorithms. That should give very good results. It is interesting to see that the value of D_2 of the ALL set can be worse than the value obtained by some algorithms. Indeed, as explained in section 3.5.2 of chapter 3, even if the ALL set weakly dominates all other sets, the indicator D_2 can be higher.

However, we show now that is possible to obtain better results than the ALL set, for the indicators considered, in reasonable time, with 2PPLS, using the neighborhood $\mathcal{N}_2$ and the MEMOTS subroutine. We have carefully selected the parameters such that we obtain better or equal results than ALL for all indicators. The initial population of 2PPLS is generated following P_3 with $S = 100$. The other parameters are the following:

- 250-2: $L = 8$ and $N = 175$.
- 500-2: $L = 15$ and $N = 100$.
- 750-2: $L = 9$ and $N = 100$.

Table 7.13: Values of the indicators obtained by state-of-the-art algorithms (1).

Instance	Algorithm	$\mathcal{H}(10^7)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$
250-2	SPEA	9.1892	1.045845	358.980899	5.322	26.549	55.57
	MOGLS00	9.8224	1.007709	259.437581	0.865	4.423	170.70
	MOGLS04	9.8633	1.002702	248.966781	0.233	2.688	202.35
	PMA	9.8621	1.003300	249.693849	0.283	2.512	169.30
	IMMOGLS	9.8556	1.005162	253.442812	0.547	3.876	82.80
	MOGTS	9.8654	1.001628	247.838096	0.147	2.225	287.00
	GRASP	9.8527	1.002634	249.350015	0.289	5.308	209.00
	MOTGA	9.7996	1.007055	252.819735	0.671	10.422	98.70
	PATH-RELINKING	9.0263	1.059320	401.867340	7.286	30.755	56.63
	ALL	9.8689	1.001067	246.297332	0.078	2.838	363.00
500-2	SPEA	36.7397	1.063298	875.672949	12.197	38.035	34.53
	MOGLS00	40.5567	1.009310	463.392108	1.679	4.133	250.50
	MOGLS04	40.7568	1.001869	435.883789	0.252	2.409	281.85
	PMA	40.7567	1.002382	436.409679	0.274	2.864	260.35
	IMMOGLS	40.7381	1.003685	441.062922	0.510	2.285	96.40
	MOGTS	40.7635	1.001632	434.491032	0.197	2.045	448.00
	GRASP	40.7169	1.001661	437.470356	0.328	1.382	357.00
	MOTGA	40.6417	1.003910	437.870310	0.498	9.125	165.75
	PATH-RELINKING	37.7366	1.047607	692.959785	9.794	40.075	96.50
	ALL	40.7850	1.000514	431.989012	0.082	2.045	671.00
750-2	SPEA	77.3235	1.086061	1738.147328	18.762	50.711	34.20
	SPEA2	83.5896	1.040974	1026.315604	7.395	40.753	250.00
	MOGLS00	88.5439	1.009544	799.795896	2.213	6.490	359.10
	MOGLS04	89.3263	1.002020	750.011634	0.294	1.494	329.25
	PMA	89.3258	1.002222	750.132256	0.301	1.512	323.45
	IMMOGLS	89.3089	1.004330	755.816782	0.524	2.134	121.60
	MOGTS	89.3134	1.003420	755.423570	0.457	1.494	545.00
	GRASP	89.1427	1.002076	755.449704	0.495	3.744	552.00
	MOTGA	89.0923	1.003735	750.392953	0.460	10.940	249.90
	PATH-RELINKING	81.7571	1.057017	1190.910247	11.992	51.920	99.10
	ALL	89.3447	1.000553	744.103070	0.093	1.494	991.00

Table 7.14: Values of the indicators obtained by state-of-the-art algorithms (2).

Instance	Algorithm	$P_{Y_N}(\%)$	$P_{Y_{SN}}(\%)$	$P_{Y_{NN}}(\%)$	$P_{Y_\triangle}(\%)$
250-2	SPEA	0.00	0.00	0.00	0.19
	MOGLS00	0.00	0.00	0.00	14.09
	MOGLS04	3.96	16.50	3.01	68.90
	PMA	2.80	13.25	2.01	63.89
	IMMOGLS	0.84	5.00	0.52	45.53
	MOGTS	9.33	25.00	8.14	78.75
	GRASP	1.58	10.00	0.95	62.68
	MOTGA	1.19	8.13	0.66	57.94
	PATH-RELINKING	0.00	0.00	0.00	1.08
	ALL	27.46	60.00	25.00	94.21
500-2	SPEA	0.00	0.00	0.00	0.00
	MOGLS00	0.00	0.00	0.00	1.70
	MOGLS04	0.32	2.27	0.21	54.64
	PMA	0.30	1.62	0.22	53.61
	IMMOGLS	0.07	0.58	0.04	41.94
	MOGTS	0.35	5.19	0.07	61.16
	GRASP	0.14	0.00	0.15	37.25
	MOTGA	0.46	1.69	0.39	54.84
	PATH-RELINKING	0.00	0.00	0.00	0.65
	ALL	5.23	20.78	4.33	84.05
750-2	SPEA	0.00	0.00	0.00	0.00
	SPEA2	0.00	0.00	0.00	0.00
	MOGLS00	0.00	0.00	0.00	0.03
	MOGLS04	0.08	0.73	0.06	46.05
	PMA	0.09	1.01	0.06	46.18
	IMMOGLS	0.07	0.64	0.04	44.15
	MOGTS	0.00	0.00	0.00	27.34
	GRASP	0.00	0.00	0.00	14.67
	MOTGA	0.07	1.19	0.02	52.12
	PATH-RELINKING	0.00	0.00	0.00	0.00
	ALL	0.86	6.42	0.65	80.02

Table 7.15: Comparison between 2PPLS and ALL based on the indicators (1).

Instance	Algorithm	$\mathcal{H}(10^7)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
250-2	2PPLS	**9.8689**	**1.000635**	**245.815180**	**0.038**	**2.622**	457.25	5.27
	ALL	**9.8689**	1.001067	246.297332	0.078	2.838	363.00	/
500-2	2PPLS	**40.7884**	**1.000262**	**430.903375**	**0.025**	**1.874**	1131.20	25.70
	ALL	40.7850	1.000514	431.989012	0.082	2.045	671.00	/
750-2	2PPLS	**89.3484**	**1.000515**	**743.524370**	**0.074**	**1.494**	1550.80	17.98
	ALL	89.3447	1.000553	744.103070	0.093	**1.494**	991.00	/

Table 7.16: Comparison between 2PPLS and ALL based on the indicators (2).

Instance	Algorithm	$P_{Y_N}(\%)$	$P_{Y_{SN}}(\%)$	$P_{Y_{NN}}(\%)$	$P_{Y_\triangle}(\%)$
250-2	2PPLS	**58.93**	**77.50**	**57.53**	**98.41**
	ALL	27.46	60.00	25.00	94.21
500-2	2PPLS	**42.51**	**73.51**	**40.72**	**98.23**
	ALL	5.23	20.78	4.33	84.05
750-2	2PPLS	**3.73**	**13.12**	**3.38**	**84.49**
	ALL	0.86	6.42	0.65	80.02

The results for 2PPLS are given in Tables 7.15 and 7.16. We see that we obtain better or equal values for all indicators, in very reasonable running times: 5s for 250-2, 26s for 500-2 and 18s for 750-2. The 2PPLS method with this configuration seems thus very competitive.

We now compare the results of MEMOTS and 2PPLS with the results of three multiobjective memetic algorithms: MOGLS [114], PMA [120] and IMMOGLS [110]. The results of these algorithms have been generated with the library MOMHLib++ [113]. The values used for the various parameters of these algorithms correspond to the default values defined in the library. We have run the algorithms on the same computer than used for running MEMOTS and 2PPLS, which allows to have a reasonably fair comparison of the running times, even if we do not use the same computer language and the same data structures.

The results are presented in Tables 7.17 and 7.18.

The results of MEMOTS have been obtained with the parameters of Table 7.4. The initial population has been generated following P_2.

For 2PPLS, the initial population has been generated following P_3 with $S = 100$, with the neighborhood $\mathcal{N}_2$ and the MEMOTS subroutine with $N = 175$ and $L = 8$ for the 250-2 instance, $N = 100$ and $L = 15$ for the 500-2 instance, and $N = 200$ and $L = 11$ for the 750-2 instance. These parameters allow to obtain comparable running times.

We see that 2PPLS allows to obtain most of the best values for the indicators, except on the 250-2 instance where MEMOTS obtains better values for $\mathcal{H}$ and D_2 and on the 750-2 instance where MEMOTS obtains better values for $\mathcal{H}$ and MOGLS04 for D_2.

Mann-Whitney test

The results of the Mann-Whitney statistical test (see chapter 3, section 3.5.4) comparing the indicators of 2PPLS with those of MOGLS04 are presented in Table 7.19. The starting level of risk of the test has been fixed to 1%.

The results show that we can affirm with a very low risk level that, for all main indicators considered, 2PPLS is better than MOGLS04 on all the biobjective instances tested, except for D_2 for which the results generated by both algorithms are incomparable.

[2] We thank Vincent Barichard for the results of MOGTS, Dalessandro Soares Vianna for the results of GRASP, Maria João Alves for the results of MOTGA and Ricardo Beausoleil for the results of PATH-RELINKING.

Table 7.17: Comparison between MEMOTS, 2PPLS and state-of-the-art algorithms on biobjective instances (1).

Instance	Algorithm	$\mathcal{H}(10^7)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
250-2	MEMOTS	**9.8699**	1.001122	246.469702	0.072	**0.822**	394.15	5.38
	2PPLS	9.8689	**1.000635**	**245.815180**	**0.038**	2.622	457.25	5.27
	MOGLS04	9.8633	1.002702	248.966781	0.233	2.688	202.35	7.00
	PMA	9.8621	1.003300	249.693849	0.283	2.512	169.30	**5.00**
	IMMOGLS	9.8556	1.005162	253.442812	0.547	3.876	82.80	6.00
500-2	MEMOTS	40.7817	1.000523	431.850981	0.073	2.235	846.85	23.32
	2PPLS	**40.7884**	**1.000262**	**430.903375**	**0.025**	**1.874**	1131.20	25.70
	MOGLS04	40.7568	1.001869	435.883789	0.252	2.409	281.85	22.00
	PMA	40.7567	1.002382	436.409679	0.274	2.864	260.35	**20.00**
	IMMOGLS	40.7381	1.003685	441.062922	0.510	2.285	96.40	23.00
750-2	MEMOTS	**89.3498**	1.000410	743.496760	0.070	1.579	1538.55	59.45
	2PPLS	89.3496	**1.000402**	**743.050365**	**0.056**	1.498	1825.55	50.72
	MOGLS04	89.3263	1.002020	750.011634	0.294	**1.494**	329.25	53.00
	PMA	89.3258	1.002222	750.132256	0.301	1.512	323.45	**50.00**
	IMMOGLS	89.3089	1.004330	755.816782	0.524	2.134	121.60	57.00

Table 7.18: Comparison between MEMOTS, 2PPLS and state-of-the-art algorithms on biobjective instances (2).

Instance	Algorithm	$P_{Y_N}(\%)$	$P_{Y_{SN}}(\%)$	$P_{Y_{NN}}(\%)$	$P_{Y_\triangle}(\%)$
250-2	MEMOTS	25.15	52.75	23.06	90.42
	2PPLS	**58.93**	**77.50**	**57.53**	**98.41**
	MOGLS04	3.96	16.50	3.01	68.90
	PMA	2.80	13.25	2.01	63.89
	IMMOGLS	0.84	5.00	0.52	45.53
500-2	MEMOTS	6.31	14.68	5.83	84.90
	2PPLS	**42.51**	**73.51**	**40.72**	**98.23**
	MOGLS04	0.32	2.27	0.21	54.64
	PMA	0.30	1.62	0.22	53.61
	IMMOGLS	0.07	0.58	0.04	41.94
750-2	MEMOTS	1.24	4.22	1.13	82.73
	2PPLS	**6.71**	**19.31**	**6.24**	**88.21**
	MOGLS04	0.08	0.73	0.06	46.05
	PMA	0.09	1.01	0.06	46.18
	IMMOGLS	0.07	0.64	0.04	44.15

Table 7.19: Comparison between 2PPLS and MOGLS04 for the biobjective instances: results of the Mann-Whitney test for the $\mathcal{H}$, $I_{\epsilon 1}$, R, D_1, D_2 and P_{Y_N} indicators.

Instance	$\mathcal{H}$	$I_{\epsilon 1}$	R	D_1	D_2	P_{Y_N}
250-2	>	>	>	>	=	>
500-2	>	>	>	>	=	>
750-2	>	>	>	>	=	>

Outperformance relations

We now compare the solutions obtained with 2PPLS and MOGLS04 in term of outperformance relations (see chapter 3, section 3.5.5).

We show in Figure 7.19 the results of the comparison between the potentially non-dominated points obtained with 2PPLS and with MOGLS04, for the biobjective instances.

These box-plots show that many solutions obtained with 2PPLS dominate solutions of MOGLS04 (about 80%) and that very few solutions obtained with 2PPLS are dominated by at least one solution of MOGLS04.

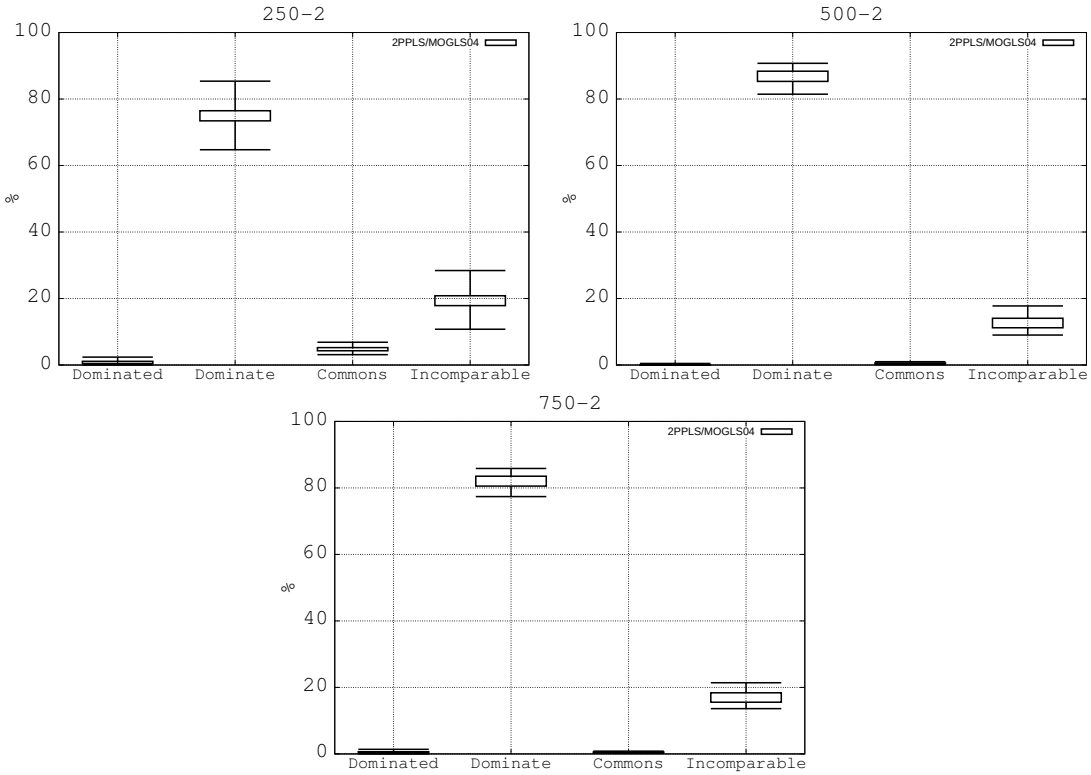

Figure 7.19: Comparison between the potentially non-dominated points obtained with 2PPLS and with MOGLS04 for the biobjective instances.

7.8.2 Three-objective instances

As said before, 2PPLS is essentially suited to solve biobjective instances. We first show it.

We have applied 2PPLS to the 250-3 instance with the following configuration. We use the way P_2 to generate the initial population. We try two neighborhoods: $\mathcal{N}_1$ and $\mathcal{N}_2$ with MEMOTS as subroutine with the following parameters: $L = 12$, $N = 200$, $t = 3$, $it_{stop} = 1$.

For the neighborhood in the MEMOTS subroutine, q is equal to 1 and β to 100. The initial population in MEMOTS is produced following P_3 with $S = 150$.

We present the results in Table 7.20 for the 250-3 instance. We also add the results obtained by MEMOTS with the parameters of Table 7.4 and the way P_2 to generate the initial population. The results of 2PPLS correspond to only one run (for computational overhead reason). The running times indicated do not include the time needed to generate the initial population.

Table 7.20: Comparison between 2PPLS and MEMOTS for the 250-3 instance (1).

Instance	Algorithm	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
250-3	2PPLS-$\mathcal{N}_1$	1.018145	306.910721	3.899	**4.798**	45947.00	35999.72
	2PPLS-$\mathcal{N}_2$	**1.017902**	**306.469352**	**3.863**	**4.798**	68540.00	28626.80
	MEMOTS	1.020264	309.680656	4.074	5.214	12043.00	**129.46**

We see that the results obtained with 2PPLS-$\mathcal{N}_2$ are of better quality for all the indicators considered. The number of potentially efficient solutions obtained with 2PPLS-$\mathcal{N}_2$ (68540 potentially efficient solutions generated!) is more than five times more important than with MEMOTS. But the running time of 2PPLS is very high: about 10 hours for 2PPLS-$\mathcal{N}_1$ and 8 hours for 2PPLS-$\mathcal{N}_2$! Indeed, the PLS method is stopped only when it is no more possible to find a new non-dominated neighbor from one of the potentially efficient solutions. In the case of these three-objective instances, there are so many potentially efficient solutions, that the stop condition is only met after a long running time.

From these results, we can say that it will be impossible to use 2PPLS to solve in a reasonable time the 500-3 and 750-3 instances, except if the method is stopped before convergence or if the neighborhood is drastically limited.

We have done that for the 250-3 instance, where we limit the running time of 2PPLS-$\mathcal{N}_2$ to the same running time than MEMOTS. The results are shown in Table 7.21, for two different running times.

Table 7.21: Comparison between 2PPLS and MEMOTS for the 250-3 instance (2).

Instance	Algorithm	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
250-3	MEMOTS	**1.025566**	**313.875552**	**4.283**	**5.989**	4581.50	8.01
	2PPLS-$\mathcal{N}_2$	1.029983	317.772372	4.363	6.932	4872.20	8.25
250-3	MEMOTS	**1.020264**	**309.680656**	4.074	**5.214**	12043.00	129.46
	2PPLS-$\mathcal{N}_2$	1.020712	311.238745	**4.034**	5.510	13504.35	129.18

We see that the results obtained with MEMOTS are better than the results of 2PPLS, except when the running time is higher, but only for the D_1 indicator. Stopping 2PPLS before convergence can indeed yield results of second-rate quality, since in PLS the exploration of the decision space is not managed as in MEMOTS where the search is constantly guided in regions of low density (in objective space).

Therefore, for the three-objective instances, we will only compare the results of MEMOTS with state-of-the-art results.

Comparison based on the mean of the indicators

The results of MEMOTS have been obtained with the parameters of Table 7.4, except the number N of iterations which is limited in order to have comparable running times with the state-of-the-art methods. The initial population has been generated following P_2. The results of the other MOMHs have been obtained with the MOMHLib++ library [113] with the default values for the parameters.

The results are presented in Table 7.22. We see that MEMOTS allows to obtain better results for all indicators considered.

Table 7.22: Comparison of MEMOTS with state-of-the-art algorithms for the three-objective instances.

Instance	Algorithm	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
250-3	MEMOTS	**1.025566**	**313.875552**	**4.283**	**5.989**	4581.50	**11**
	MOGLS04	1.028215	315.637081	4.384	6.140	1928.95	16
	PMA	1.030180	316.623463	4.429	6.457	1635.75	**11**
	IMMOGLS	1.032501	324.902766	4.840	7.412	483.65	**11**
500-3	MEMOTS	**1.021406**	**595.645516**	**4.794**	**7.062**	8224.45	**38**
	MOGLS04	1.022422	599.213696	4.938	7.359	2827.20	49
	PMA	1.022247	599.326989	4.933	7.198	2755.95	**38**
	IMMOGLS	1.027760	613.902801	5.526	9.572	450.45	40
750-3	MEMOTS	**1.019268**	**863.188041**	**5.445**	**8.021**	13420.35	**95**
	MOGLS04	1.020150	868.795664	5.610	8.189	2930.15	104
	PMA	1.019519	868.529211	5.600	8.096	2900.95	**95**
	IMMOGLS	1.024708	884.603911	6.031	10.041	465.30	98

Mann-Whitney test

The results of the Mann-Whitney statistical test are shown in Table 7.23. The results show that we can affirm with a very low risk that, for the indicators considered in this work, the results obtained with MEMOTS are better or equal than MOGLS04 on all three-objective instances tested in this work.

Table 7.23: Comparison between MEMOTS and MOGLS04 for the three-objective instances: results of the Mann-Whitney test for the $I_{\epsilon 1}$, R, D_1 and D_2 indicators.

Instance	$I_{\epsilon 1}$	R	D_1	D_2
250-3	$>$	$>$	$>$	$=$
500-3	$>$	$>$	$>$	$=$
750-3	$=$	$>$	$>$	$=$

Outperformance relations

The results of the outperformance relations are presented in Figure 7.20. We see that about 30% of the solutions obtained with MEMOTS dominate solutions of MOGLS04 and that about 20% of the solutions obtained with MEMOTS are dominated by at least one solution of MOGLS04. There are also many incomparable solutions (about 50%).

7.9 Conclusion and perspectives

In this chapter, we have shown how to adapt the 2PPLS and MEMOTS methods to the resolution of the MOMKP. On biobjective instances, 2PPLS with the neighborhood that uses a simplified version of the MEMOTS method as a subroutine gives better results than the stand-alone MEMOTS method. On the other hand, on three-objective instances, the convergence time of 2PPLS is very high and MEMOTS turns out to be more efficient.

We have compared the results with state-of-the-art algorithms, and we have shown that we obtain better results for the indicators considered.

Among the perspectives, to experiment the methods on another type of instances would be challenging.

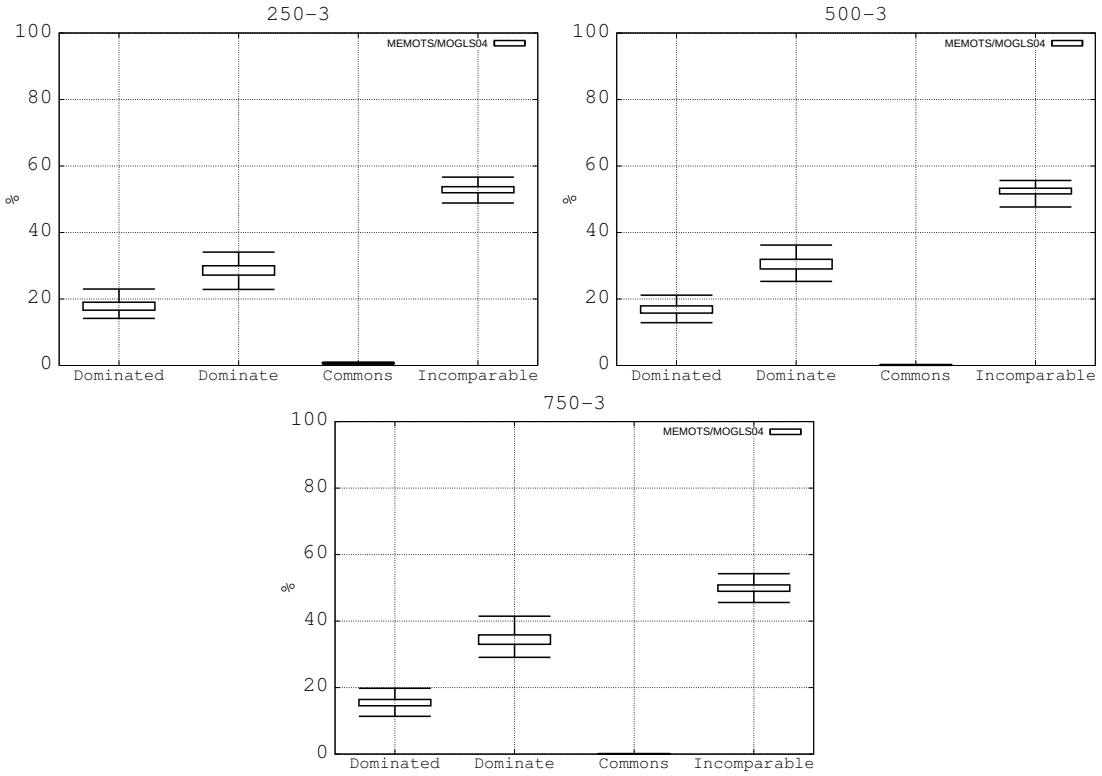

Figure 7.20: Comparison between the potentially non-dominated points obtained with MEMOTS and with MOGLS04 for the three-objective instances.

We have however pointed out that the number of efficient solutions of the three-objective instances considered is very high (we have generated more than 68500 different solutions for the 250-3 instance, it would not be surprising that the efficient set contains more than 100000 solutions) and it would be very hard to develop an efficient exact method for this kind of problems.

We think that in the case of randomly generated instances of the MOMKP with more than two objectives, the decision maker should get involved in the process of generating solutions, at the beginning or during the execution of the algorithm, in order to direct the search and to limit the number of solutions generated. An a posteriori approach should be avoided as often as possible. Indeed, if 2PPLS revealed very competitive on biobjective instances, on three-objective instances, as the search in 2PPLS is not as well-directed than in MEMOTS or than in the other memetic algorithms using scalarizing functions to guide the search, the results were not so impressive. That corroborates the need of the intervention of the decision maker during the process of 2PPLS.

8

Resolution of the bTSP with 2PPLS

From the preceding survey about the MOMHs for the resolution of the MOTSP (see chapter 5), we have seen that many methods, using sometimes complex components, have been developed for approximating X_E. We show here that it is however possible to produce very good approximations $\widetilde{X}_E$, better for several indicators than state-of-the-art algorithms, for different types of instances of the MOTSP, with a simple adaptation of 2PPLS.

In this chapter, only 2PPLS has been applied to the biobjective traveling salesman problem (bTSP). We do not adapt MEMOTS given that we have previously shown that 2PPLS gives better results than MEMOTS on the biobjective instances of the MOMKP. Moreover, we will use here an adaptation of 2PPLS that does not require any numerical parameters, contrary to MEMOTS.

This chapter is organized as follows. We first present how 2PPLS has been adapted to the bTSP (section 8.1). The data and reference sets are described in section 8.2. The study of the different alternatives for 2PPLS is presented in section 8.3. The results and a comparison with state-of-the-art algorithms are exposed in section 8.4. In section 8.5, we present a simple way to improve the quality of the results obtained with 2PPLS by using a single-objective solver with a data perturbation technique. However, we will see that it would be impossible to solve large-scale instances of the bTSP in reasonable running times with 2PPLS as it is. We therefore show in section 8.6 how the traditional speed-up techniques for the single-objective TSP can be adapted to the bTSP and integrated into 2PPLS[1]. In section 8.6.4 we analyse the results obtained by the different speed-up techniques and in section 8.6.5 we realize a comparison between the speed-up techniques. Finally, in section 8.6.6, we compare the results of 2PPLS with speed-up techniques to state-of-the-art results.

As a reminder, the bTSP is defined as follows:

Given n cities and two costs $c_k(i,j)\,(k=1,2)$ to travel from city i to city j, the MOTSP consists in finding a tour, that is a Hamiltonian cycle δ of the n cities, minimizing

$$\text{“min”}\, f_k(\delta) = \sum_{i=1}^{n-1} c_k(\delta(i),\delta(i+1)) + c_k(\delta(n),\delta(1)),\ \ k=1,2$$

We only consider the symmetric bTSP, that is $c_k(i,j) = c_k(j,i)$.

8.1 Adaptation of 2PPLS to the bTSP

2PPLS only requires two elements to be adapted to a MOCO problem: an initial population and a neighborhood. In both cases, we will use simple elements. We will thoroughly make use of the Lin-Kernighan heuristic [152] and two-edge exchange moves.

[1] This particular work has been realized in collaboration with Professor A. Jaszkiewicz during our stay at Politechnika Poznańska (Poland).

8.1.1 Initial population

For the initial population, we will try to generate a good approximation of the supported efficient solutions. With this aim, we use the heuristic adaptation of the dichotomic method of Aneja and Nair presented in chapter 6, section 6.3.1. As a reminder, the method consists in generating all weight sets which make it possible to obtain a minimal complete set of extreme supported efficient solutions of a biobjective problem. Each single-objective problem resulting from the weighted sum is solved with a single-objective heuristic. Here, the single-objective problems are solved with one of the best heuristics for the single-objective TSP: the Lin-Kernighan heuristic (LK). The LK heuristic was initially proposed by Lin and Kernighan in 1973 [152], and is always considered as one of the most effective methods for generating near-optimal solutions for the single-objective symmetric TSP. Generalization of two-edge and three-edge exchange moves are considered in this heuristic.

In this work, we use two very efficient versions of the LK heuristic, with a published implementation. Both improve the original version of Lin and Kernighan. The two versions are the following:

- The version implemented by Helsgaun [103], that we call LKH. The code source of this method has been published on `http://www.akira.ruc.dk/~keld`.
- The chained Lin-Kernighan version of Applegate *et al.* [7], the code source of this method has been published on `http://www.tsp.gatech.edu/concorde` through the Concorde package, that includes various methods for solving exactly or heuristically the single-objective TSP. We call this method LKC.

In each solver, we use the default parameters given by the authors, except for the LKH solver, where the maximum number of candidate edges associated to each node (that is the maximum number of edges that are taken into account for a move from a node) has been fixed to 3 in the place of 5 [103] for the instances of more than 200 cities, in order to reduce the running time of the first phase.

8.1.2 Neighborhood

For the neighborhood needed in PLS, we use the well-known two-edge exchange neighborhood (see Figure 8.1), as suggested by Borges and Hansen in their work about the global convexity in the MOTSP [23]: efficient solutions of the MOTSP are very close in the decision space and two-edge exchange moves from efficient solutions allow to generate many other efficient solutions.

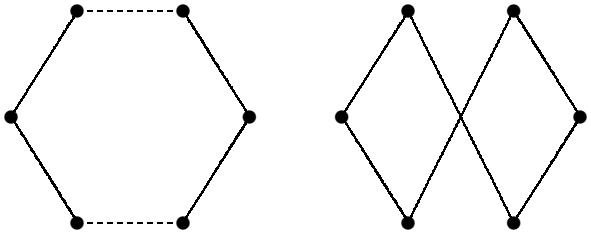

Figure 8.1: Illustration of the two-edge exchange neighborhood.

We will also compare the two versions of PLS, presented in chapter 3, section 3.4.1. The version 1, called PLS1, is the version of Angel *et al.* [6]. The version 2, called PLS2, is the version of Paquete *et al.* [184], where the population containing the potentially efficient solutions whose the neighborhood has not yet been exploited, is immediately updated after the addition of a neighbor. The results of PLS2 depend on the order according to which the solutions of the population are examined. The rule is to take the first solution of the population, that is the solution that minimizes the first objective, given that the solutions of the population are continuously sorted according to the first objective in order to speed up the `AddSolution` procedure (see section 3.4 in chapter 3).

8.2 Data and reference sets

We test 2PPLS on several biobjective TSP instances. We consider four types of instances:

- Euclidean instances: the costs between the edges correspond to the Euclidean distance between two points in a plane.
- Random instances: the costs between the edges are randomly generated from a uniform distribution.
- Mixed instances: the first cost corresponds to the Euclidean distance between two points in a plane and the second cost is randomly generated from a uniform distribution.
- Clustered instances: the points are randomly clustered in a plane, and the costs between the edges correspond to the Euclidean distance.

The following biobjective symmetric instances are used:

- Euclidean instances: six Euclidean instances of 100 cities (KroAB100, KroAC100, KroAD100, KroBC100, KroBD100, KroCD100), one instance of 150 cities (KroAB150) and one instance of 200 cities (KroAB200). These instances have been generated on the basis of single-objective TSP instances of the TSPLIB library [198]. State-of-the-art results have been published for these instances. Besides, we have generated new instances, called KroAB300, KroAB400, KroAB500, KroAB750 and KroAB1000, by randomly generating coordinates. We will only use these instances later in this chapter. We also use three more Euclidean instances of respectively 100, 300 and 500 cities called EuclAB100, EuclAB300 and EuclAB500, generated and published by Paquete [183].
- Random instances: three random instances of respectively 100, 300 and 500 cities called RdAB100, RdAB300 and RdAB500, generated and published by Paquete [183].
- Mixed instances: three mixed instances of respectively 100, 300 and 500 cities called MixedAB100, MixedAB300 and MixedAB500, also generated and published by Paquete [183].
- Clustered instances: three clustered instances of respectively 100, 300 and 500 cities called ClusteredAB100, ClusteredAB300 and ClusteredAB500. We have generated ourselves the clustered instances with the generator available from the 8th DIMACS Implementation Challenge site[2] with the following property: the number of clusters is equal to the number of cities divided by 25 and the maximal coordinate for a city is equal to 3163 (as done by Paquete for the Euclidean instances).

To compute the D_1, D_2 and $I_{\epsilon 1}$ indicators (see section 3.5.1 in chapter 3), we need reference sets. Here, the reference sets have been generated on the basis of the notion of ideal set introduced in section 3.5.3 of chapter 3. As a reminder, the ideal set is defined as the best potential Pareto front that can be produced from the extreme supported non-dominated points. This is a lower bound of the Pareto front [57]. For generating the extreme supported non-dominated points, we have used the method proposed by Przybylski *et al.* [193] (see section 3.2.3 in chapter 3). As exact solver for this method, we use the exact single-objective solver provided by the Concorde package. However, for the instances of more than 200 cities, numerical problems were encountered. Indeed, in order to use the single-objective solver, we create a single-objective instance of the TSP by merging the two matrices of cost associated to a biobjective instance. This merge is done thanks to a weighted sum. The problem is that the weight set can be very large, and the values of the costs of the new single-objective instances may be too high to be processed by the exact solver of Concorde (which only accepts integer values limited to 2^{32}).

Thus, for these instances, we have generated the extreme supported non-dominated points of the biobjective minimum spanning tree problem (bMST) associated to the same data than

[2] `http://www.research.att.com/~dsj/chtsp/download.html`

the bTSP. The ideal set is then produced on the basis of the extreme supported non-dominated points of the bMST. As the biobjective minimum spanning tree problem is a relaxation of the bTSP, all feasible solutions of the bTSP are dominated or equivalent to the solutions of the ideal set of the bMST. The MST has the advantage, in opposition to the TSP, to be easily solved in polynomial time by using the algorithms of Kruskal [141] or Prims [192].

For the computation of the R indicator, the reference points y^0 are determined according to the reference sets (they are equal to the local ideal points of these reference sets), except for the "Kro" instances with 200 cities or less, where the reference points can be found in [118] for the 100 cities instances and in [187] for the KroAB150 and KroAB200 instances. The number of weight sets used is equal to 101 for all instances. This indicator has been normalized between 0 and 1 (where 1 corresponds to the best value achievable), in accordance with the formula (3.6) of section 3.5.1 in chapter 3. We have realized this normalization because previous authors did, and therefore that allows us to compare our results on the basis of the values of this indicator. As normalization is done, a bounding point y^1 is needed, which is also necessary for the computation of the hypervolume. The bounding points y^1 are obtained by multiplying by 1.1 the coordinates of the local nadir points of the reference sets, except for the "Kro" instances with 200 cities or less, where the bounding points can be found in [114] for the 100 cities instances and in [187] for the KroAB150 and KroAB200 instances. We multiply by 1.1 in order to avoid that a feasible point does not dominate the bounding point (see section 3.5.1 in chapter 3), which is particularly problematical for the computation of the hypervolume.

All the algorithms tested in this chapter have been run on a Pentium IV with 3 Ghz CPUs and 512 MB of RAM. We present the average of the indicators on 20 executions. The running time of our implementation of the algorithms corresponds to the wall clock time.

8.3 Study of the different alternatives for 2PPLS

In this section, we first study the influence of the way to generate the initial population (section 8.3.1) and after we compare the two versions of PLS used as second phase in 2PPLS (section 8.3.2).

8.3.1 Initial population

We first compare the results obtained by the different solvers considered in the dichotomic method of Aneja and Nair. We then study the influence of the size of the initial population.

Influence of the solver

We compare here three different solvers used in the dichotomic method of Aneja and Nair for the generation of the initial population. The first solver is LKC, the second is LKH and the last is the exact method coming from the Concorde package. We only apply the first phase of 2PPLS.

We can see the results in Table 8.1 under the names Di-LKC for LKC, Di-LKH for LKH and Di-EXA for the exact method.

Four instances of 100 cities are considered, of different types: kroAB100, RdAB100, MixedAB100 and ClusteredAB100, one instance of 150 cities: KroAB150, and one instance of 200 cities: KroAB200. We use the following indicators to assess the quality of the approximations obtained: the D_{1SE}, D_{2SE} and $I_{\epsilon 1SE}$ indicators that have the same meaning than the D_1, D_2 and $I_{\epsilon 1}$ indicators presented in section 3.5.1 of chapter 3, but as we are seeking in the first phase an approximation of a minimal complete set of extreme supported efficient solutions, we use the exact minimal complete set of extreme supported efficient solutions given by the exact method as reference set. We also indicate the percentage $P_{Y_{SN1}}$ of extreme non-dominated points generated and as additional information, the number $|PE|$ of potentially efficient solutions obtained and the running time in seconds. The conclusions of this analysis are as follows:

- The LKH solver gives better results than the LKC solver. The differences in the values of the indicators are not large, but the differences are accentuating when the size of the instance increases. On the other hand, the running time with LKC solver is lower, except for the RdAB100 and ClusteredAB100 instances.
- Presumably the most important result is that the quality of the approximations obtained with the heuristic method is very good. With the LKH solver, we find, at worst 79.95% of the solutions belonging to the extreme supported non-dominated points set (for the KroAB200 instance). For the instances of 100 cities, the worst results were obtained for the ClusteredAB100 instance (87.48%) which is still good. The saving in running time in term of ratio with LKH in comparison with the exact method goes from 7 (ClusteredAB100 instance) to 24 (KroAB100 instance).

We can also remark that the number of potentially efficient solutions generated by a heuristic method can be higher than the number of extreme efficient solutions generated by the exact method. It is because, with a heuristic method, some potentially non-supported efficient solutions can be sometimes generated.

Table 8.1: Comparison of the LKC and LKH solvers for the first phase.

Instance	Algorithm	$I_{\epsilon 1SE}$	D_{1SE}	D_{2SE}	$P_{Y_{SN1}}(\%)$	$\|PE\|$	Time(s)
	Di-EXA	1.000000	0.000	0.000	100.00	109.40	772.09
KroAB100	Di-LKC	1.025895	0.342	**6.867**	88.21	109.60	**27.77**
	Di-LKH	**1.019012**	**0.197**	7.158	**91.65**	110.85	31.85
	Di-EXA	1.000000	0.000	0.000	100.00	164.00	1226.90
KroAB150	Di-LKC	1.024320	0.622	11.746	78.29	160.60	**59.46**
	Di-LKH	**1.021176**	**0.406**	**9.318**	**84.63**	154.90	93.21
	Di-EXA	1.000000	0.000	0.000	100.00	222.00	2606.02
KroAB200	Di-LKC	1.024008	0.709	10.451	65.61	216.85	**105.76**
	Di-LKH	**1.017208**	**0.452**	**9.025**	**79.95**	212.40	186.33
	Di-EXA	1.000000	0.000	0.000	100.00	77.00	217.02
RdAB100	Di-LKC	1.049386	1.015	13.786	76.95	87.55	25.28
	Di-LKH	**1.012554**	**0.117**	**5.832**	**96.56**	76.95	**22.76**
	Di-EXA	1.000000	0.000	0.000	100.00	98.00	358.28
MixedAB100	Di-LKC	1.029337	0.607	8.474	80.97	106.00	**25.97**
	Di-LKH	**1.025061**	**0.351**	**7.988**	**92.24**	93.00	26.57
	Di-EXA	1.000000	0.000	0.000	100.00	109.00	185.61
ClusteredAB100	Di-LKC	1.044280	1.251	17.072	76.24	94.00	48.49
	Di-LKH	**1.020360**	**0.379**	**9.156**	**87.48**	104.60	**25.41**

As we can see there are differences in the results between LKC and LKH although both solvers are known to be able to find the optimum of small single-objective TSP problems. However, for some weight sets, it seems that "difficult" single-objective instances of the TSP are generated. We have represented in Figure 8.2 the histogram showing the repartition of the running time needed to solve with the exact method the single-objective instances obtained with the dichotomic method, for the KroAB100 instance. We remark that about 75% of the single-objective instances resulting from KroAB100 are easy to solve while there are some instances that are more difficult since the running time is much higher. That was also highlighted by Borges and Hansen in [23]. The differences between LKC and LKH are thus probably accentuated on these difficult instances.

In Table 8.2, we show the results obtained with the Di-LKC and Di-LKH algorithms for the instances of more than 200 cities, that is the EuclAB300, EuclAB500, MixedAB300, MixedAB500, RdAB300, RdAB500, ClusteredAB300 and ClusteredAB500 instances. For theses instances, we

use as reference set the ideal set based on the bMST (see section 8.2). We use five different indicators to measure the quality of the approximations obtained: the $\mathcal{H}$, $I_{\epsilon 1}$, R, D_1 and D_2 indicators. We also add as additional information the number $|PE|$ of potentially efficient solutions generated and the running time in seconds.

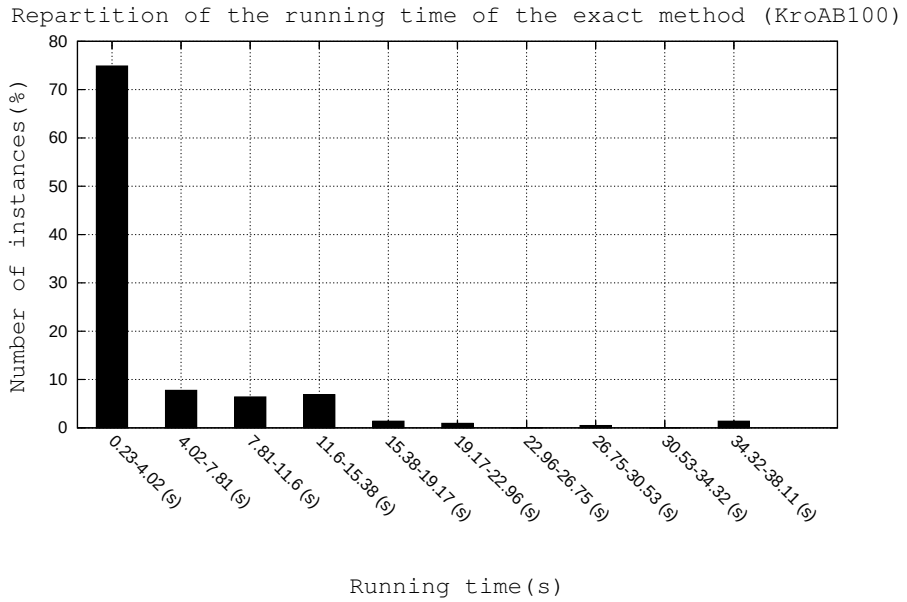

Figure 8.2: Repartition of the running time of the exact method to solve single-objective TSP instances obtained with the dichotomic method of Aneja and Nair.

Table 8.2: Comparison of the LKC and LKH solvers on bigger instances for the first phase.

Instance	Algorithm	$\mathcal{H}(10^8)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
EuclAB300	Di-LKC	**2306.3557**	**1.125952**	**0.941247**	**17.044**	**22.008**	271.35	185.29
	Di-LKH	2304.6462	1.133814	0.941132	17.240	25.563	178.20	**97.26**
EuclAB500	Di-LKC	**7158.1360**	**1.129153**	**0.948687**	**17.477**	**24.061**	301.00	385.99
	Di-LKH	7156.6196	1.130715	0.948667	17.544	26.200	269.00	**367.58**
RdAB300	Di-LKC	4802.6836	1.728705	0.966741	21.088	36.425	236.60	303.19
	Di-LKH	**4804.4059**	**1.719164**	**0.966868**	**20.814**	**29.448**	285.20	**205.39**
RdAB500	Di-LKC	13983.9830	1.790381	0.973544	19.766	**55.228**	260.25	**804.66**
	Di-LKH	**13990.4166**	**1.771297**	**0.973761**	**19.205**	59.552	472.15	848.71
MixedAB300	Di-LKC	**3407.4891**	1.778788	**0.956197**	**20.371**	**31.135**	240.65	225.96
	Di-LKH	3407.1137	**1.767418**	0.956167	20.384	34.223	228.15	**149.14**
MixedAB500	Di-LKC	10433.7414	1.710660	0.960407	23.088	**32.256**	292.00	700.75
	Di-LKH	**10435.5267**	**1.691078**	**0.960507**	**22.949**	33.371	426.15	**669.68**
ClusteredAB300	Di-LKC	2559.5979	**1.244220**	**0.956072**	18.607	**34.272**	177.10	215.85
	Di-LKH	**2559.9117**	1.246607	0.956043	**18.514**	43.553	137.50	**72.98**
ClusteredAB500	Di-LKC	8502.1046	1.204523	**0.962036**	18.280	80.133	233.05	384.08
	Di-LKH	**8508.5006**	**1.182572**	0.962032	**17.470**	**40.188**	228.55	**294.83**

We remark than none of the two solvers is better than the other on all instances. On the other hand, the running time with LKH is lower than the running time with LKC on all instances, except on the RdAB500 instance.

Influence of the size of the initial population

We study here the influence of the size of the initial population on the results of 2PPLS for the Euclidean instances. To do that, we use the Di-LKC alternative for the first phase, and we limit the number of potentially efficient solutions generated by Di-LKC to m. We vary the value of the parameter m from 1 to the maximum number of potentially efficient solutions obtained with the Di-LKC alternative. The m solutions of the initial population are always taken in the order given by the dichotomic method of Aneja and Nair. For the second phase, we use the first version of PLS with the two-edge exchange neighborhood.

We can see the results in Figure 8.3 for the kroAB100 instance. We have represented the influence of m on the D_1 and $I_{\epsilon 1}$ indicators, and on the number $|PE|$ of potentially efficient solutions generated for only one execution of 2PPLS. We have also represented the influence of m on the running time of phase 1 (P1) (in red), on the running time of PLS (in green) and on the running time of 2PPLS (equal to the sum of the running times of P1 and PLS) (in blue).

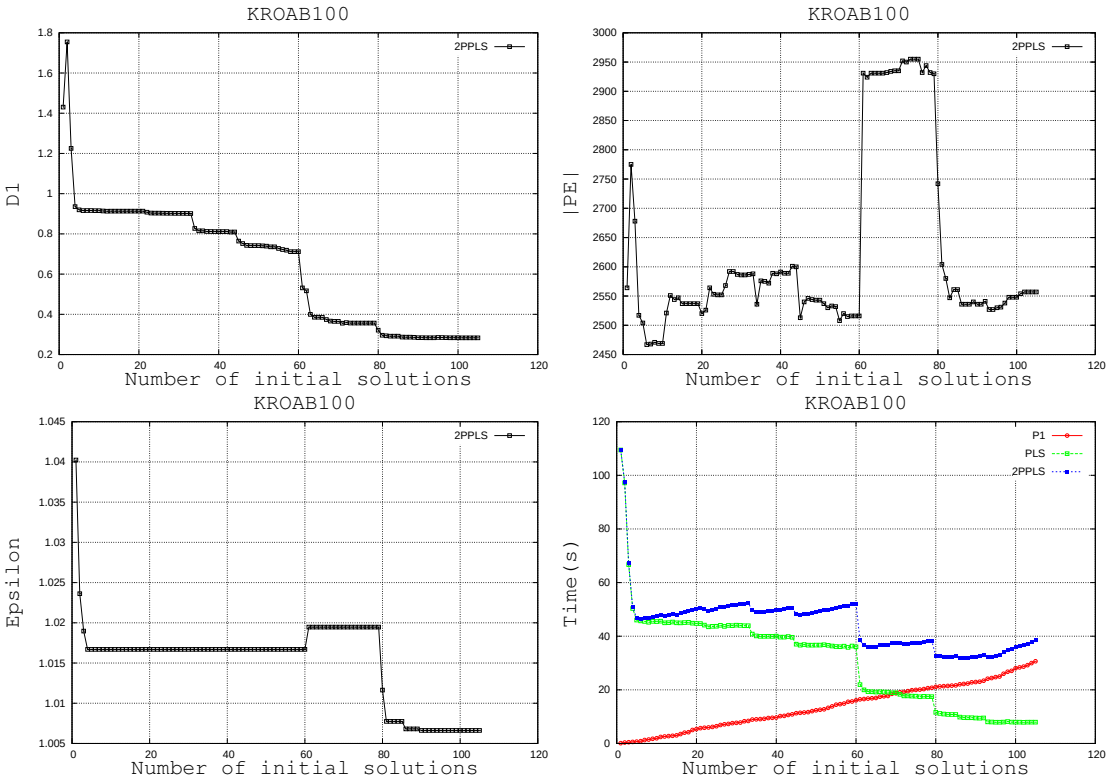

Figure 8.3: Influence of the number of initial solutions for the KroAB100 instance: dichotomic order.

We notice that by increasing m, the D_1 and $I_{\epsilon 1}$ indicators are generally improving. But there are exceptions: the second solution added increases the D_1 indicator (but reduces the $I_{\epsilon 1}$ indicator) and the 61^{th} solution increases the $I_{\epsilon 1}$ indicator (but reduces the D_1 indicator).

Another interesting observation is that the improvements are done by step, of more or less important values. For instance, the 61^{th} solution makes it possible to strongly improve the D_1 indicator of 2PPLS and to increase the number of potentially efficient solutions generated. On the other hand, there are many solutions generated during the first phase that do not allow to make improvements.

Let us define a cluster (or a connected component) as a set of solutions that are reachable from each other by applying k elementary moves with respect to a given neighborhood [188] (see section 2.2.3 in chapter 2). We can thus say that some solutions generated during the first phase allow to attain a new cluster of potentially efficient solutions by applying elementary moves, and

other new solutions of the first phase are not worthwhile because they are already connected (that is in the same cluster) to previously generated solutions .

If we look at the $|PE|$ information, we can see that, for instance, the 61^{th} solution allows to generate a new cluster of potentially efficient solutions, since the number of potentially efficient solutions is increasing following the addition of this solution. But most of the solutions of this cluster are then eliminated following the addition of a new solution generated during the first phase.

It seems thus that it is possible to improve the initial population generated during the first phase by using a method that allows to generate non-connected solutions with respect to the neighborhood used.

For the running times, we remark that the running time of the first phase is linearly increasing, but that the running time of PLS is decreasing, and always by step. The total running time of 2PPLS presents thus many local optima, and the global optimum is attained when about 85 solutions are generated. It is thus interesting, especially if the running time of 2PPLS is critical. Therefore, for instances of more than 100 cities, where the resolution of 2PPLS becomes important, it seems that is possible to reduce it by generating the best initial population according to the running time.

In Figure 8.4, we compare the results for the indicator D_1 between the same initial population but in the first possibility, the m solutions are taken in the order given by the dichotomic order and in the other possibility, the m solutions are m well-dispersed solutions of the population, in the objective space.

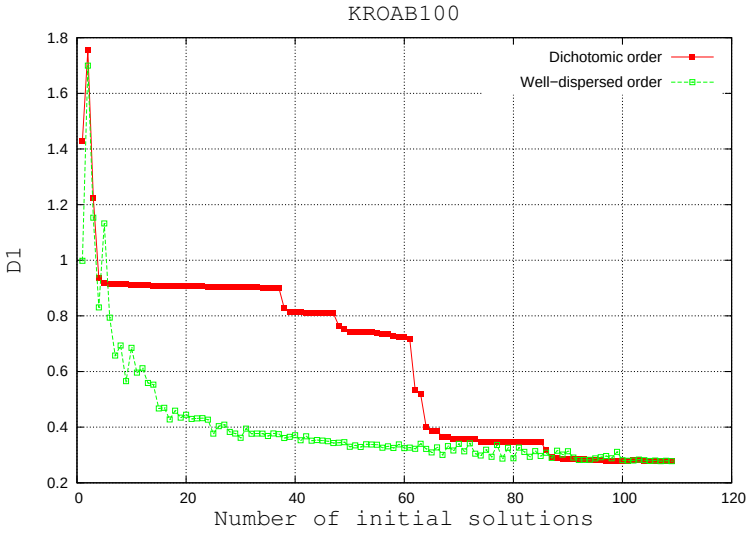

Figure 8.4: Influence of the number of initial solutions for the KroAB100 instance: comparison between the dichotomic order and the well-dispersed order.

We see that the well-dispersed order gives better results, that is for a same number of initial solutions, the indicator D_1 is lower, except for a low or high number of solutions (which is logical since in this case the same initial solutions are used).

In Figure 8.5, we compare the well-dispersed order, with the same order but in this case, only the solutions that are not in the same cluster than previous ones are taken into account (the neighborhood to define the clusters is the two-edge exchange neighborhood).

We remark that using only solutions that are located in different clusters is not very worthwhile. It is only at the end that the results with the clusters become very slightly better since we need less initial solutions to produce the same quality results.

That means that all first solutions of the initial population are all located in different clusters,

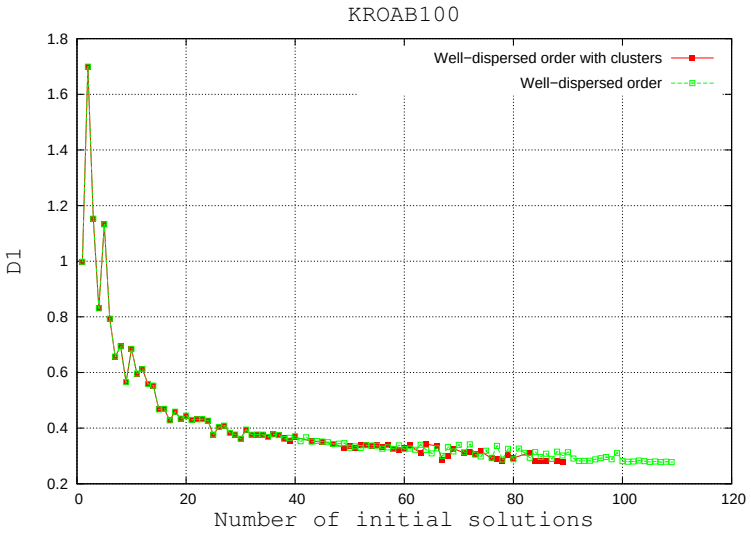

Figure 8.5: Influence of the number of initial solutions for the KroAB100 instance: comparison between the well-dispersed order and the well-dispersed order with clusters.

and only a few are in the same clusters. Moreover, generating initial solutions located in different clusters would be quite difficult.

8.3.2 Second phase

We experiment here the two different versions of PLS (PLS1 and PLS2) presented in section 3.4.1 of chapter 3. We start both versions from four different populations: a solution randomly generated (Rd) as done by Angel *et al.* [6] and Paquete *et al.* [184] and from three other populations generated by the dichotomic method with the LKC solver, the LKH solver and the exact method. Therefore, for each instance, eight methods are experimented.

We can see the results in Table 8.3 for the KroAB100, MixedAB100, RdAB100 and ClusteredAB100 instances. The running time indicated in the table corresponds only to the running time of the second phase.

If we first compare the results of the two different versions of PLS, we can draw the following conclusions:

- If PLS is run from a random solution, for the four instances, PLS1 takes more time but is better in term of quality, which is logical given that in PLS1, dominated solutions are sometimes stored in the population.
- If PLS is run from a good initial population (Di-LKC, Di-LKH or Di-EXA), the differences between the two versions of PLS in term of running time and quality are not so marked. Regardless, for the four instances, we find slightly better results with PLS1, in similar running times.
- The results when PLS is run from a good initial population are much better in term of quality and running time than if PLS is run from a random solution.

If we now compare the results given by the different populations as starting point of the PLS method, we can say that:

- The results given by the three different populations are very similar.
- To use the exact method—or using a better solver than the ones used in this work or with better parameters than the default parameters—will not have an important impact on the quality results of 2PPLS, for the 100 instances tested in this work.

Table 8.3: Comparison of PLS1 and PLS2 for the second phase, run from different initial populations, for the instances of 100 cities.

Instance	Algorithm	$\mathcal{H}(10^8)$	$I_{\epsilon 1}(SE)$	R	D_1	D_2	$\|PE\|$	Time PLS(s)
KroAB100	RD+PLS1	224.8358	1.045446	0.934207	1.461	30.154	2373.35	159.80
	RD+PLS2	224.3934	1.078011	0.934066	1.676	45.438	2521.35	22.00
	Di-LKC+PLS1	**226.1059**	**1.006617**	**0.935259**	**0.280**	2.892	2541.70	7.80
	Di-LKC+PLS2	226.1031	1.006641	**0.935259**	0.281	2.892	2494.15	7.19
	Di-LKH+PLS1	226.1039	**1.006617**	0.935255	0.283	**2.884**	2559.30	6.75
	Di-LKH+PLS2	226.0998	**1.006617**	0.935252	0.289	**2.884**	2493.85	7.22
	Di-EXA+PLS1	226.1049	**1.006617**	0.935255	0.282	2.896	2548.00	**6.72**
	Di-EXA+PLS2	226.1004	**1.006617**	0.935252	0.288	2.896	2495.00	7.16
RdAB100	RD+PLS1	517.0968	1.464269	0.948719	7.220	111.929	704.90	76.89
	RD+PLS2	515.6188	1.669674	0.948497	8.302	143.237	626.15	7.40
	Di-LKC+PLS1	**529.8228**	**1.021462**	0.953365	**0.810**	**14.170**	515.75	1.06
	Di-LKC+PLS2	529.8215	1.021510	0.953367	**0.810**	**14.170**	515.15	1.20
	Di-LKH+PLS1	529.8154	1.022070	**0.953374**	0.824	14.654	499.45	1.04
	Di-LKH+PLS2	529.8122	1.022070	0.953370	0.827	14.654	497.65	1.13
	Di-EXA+PLS1	529.8131	1.022070	0.953372	0.829	14.297	491.00	**1.00**
	Di-EXA+PLS2	529.8102	1.022070	0.953369	0.832	14.297	491.00	1.12
MixedAB100	RD+PLS1	327.2757	1.383581	0.934599	3.617	60.439	1080.20	112.81
	RD+PLS2	325.8543	1.529007	0.933849	4.747	84.643	1057.05	9.78
	Di-LKC+PLS1	331.9920	1.015601	0.936814	0.529	5.768	882.55	2.31
	Di-LKC+PLS2	**331.9926**	1.015432	**0.936818**	**0.527**	5.768	870.15	2.32
	Di-LKH+PLS1	331.9848	**1.013461**	0.936801	0.547	**4.486**	873.75	2.33
	Di-LKH+PLS2	331.9861	**1.013461**	0.936806	0.546	**4.486**	861.70	2.33
	Di-EXA+PLS1	331.9918	**1.013461**	0.936807	0.539	4.522	876.00	**2.29**
	Di-EXA+PLS2	331.9922	**1.013461**	0.936812	0.540	4.522	862.00	2.31
ClusteredAB100	RD+PLS1	266.3483	1.027901	0.949188	1.129	27.539	2464.05	183.59
	RD+PLS2	265.7003	1.055536	0.948876	1.502	33.459	2333.25	19.14
	Di-LKC+PLS1	267.3018	1.007167	0.950017	0.229	4.937	2490.30	7.19
	Di-LKC+PLS2	267.2991	1.007390	0.950017	0.233	4.937	2435.90	7.70
	Di-LKH+PLS1	267.3141	1.006992	**0.950027**	**0.218**	**2.773**	2489.80	7.06
	Di-LKH+PLS2	267.3113	**1.006983**	0.950025	0.223	**2.773**	2436.75	7.47
	Di-EXA+PLS1	**267.3147**	1.007068	**0.950027**	**0.218**	**2.773**	2473.00	**6.88**
	Di-EXA+PLS2	267.3122	1.007059	0.950026	0.222	**2.773**	2433.00	7.49

Table 8.4: Comparison of PLS1 and PLS2 for the second phase, run from different initial populations, for the instances of 300 or 500 cities.

Instance	Algorithm	$\mathcal{H}(10^8)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time PLS(s)
EuclAB300	Di-LKC+PLS1	**2309.6866**	**1.124524**	**0.941704**	**16.865**	**21.612**	14749.60	**597.85**
	Di-LKH+PLS1	2309.3302	1.127174	0.941690	16.895	23.318	14700.10	809.67
EuclAB500	Di-LKC+PLS1	**7165.3936**	**1.128923**	**0.948997**	**17.244**	21.616	34861.95	**5923.11**
	Di-LKH+PLS1	7164.9738	1.129983	**0.948997**	**17.244**	**21.596**	34075.50	6690.61
MixedAB300	Di-LKC+PLS1	3410.4396	1.778754	0.956496	20.132	**28.411**	5633.70	**221.10**
	Di-LKH+PLS1	**3410.5593**	**1.767372**	**0.956507**	**20.093**	30.435	5394.85	257.20
MixedAB500	Di-LKC+PLS1	10440.4057	1.710616	0.960654	22.856	31.708	13636.70	**1981.27**
	Di-LKH+PLS1	**10441.6266**	**1.691066**	**0.960697**	**22.712**	**31.487**	13207.50	2353.12
RdAB300	Di-LKC+PLS1	4804.5809	1.728526	0.966936	20.902	36.295	2006.55	48.08
	Di-LKH+PLS1	**4805.7635**	**1.719164**	**0.967015**	**20.712**	**29.443**	1962.05	**44.42**
RdAB500	Di-LKC+PLS1	13988.0537	1.790178	0.973703	19.494	**54.692**	3492.85	291.26
	Di-LKH+PLS1	**13992.2169**	**1.771297**	**0.973834**	**19.128**	59.552	3189.55	**221.44**
ClusteredAB300	Di-LKC+PLS1	2565.4582	1.242723	**0.956624**	17.751	**22.229**	16885.00	**1338.84**
	Di-LKH+PLS1	**2565.7109**	**1.241761**	0.956621	**17.748**	29.524	16303.10	1362.34
ClusteredAB500	Di-LKC+PLS1	8516.6889	1.183202	**0.962417**	**16.959**	24.443	42743.25	13111.25
	Di-LKH+PLS1	**8516.7852**	**1.182158**	0.962403	16.990	**23.440**	42433.30	**10237.95**

The results for the bigger instances are shown in Table 8.4, but here, we only test the difference of results obtained by 2PPLS according to the initial population: Di-LKC and Di-LKH are compared. This is always the first version of PLS (PLS1) that is used.

For these bigger instances, again, we do not find high differences in term of quality and running time between the two different populations experimented. Only for two instances one of the initial population gives better results on all indicators (for EuclAB300, Di-LKC and for RdAB300 instance, Di-LKH).

8.4 Comparison with state-of-the-art results

8.4.1 Comparison based on the mean of the indicators

Kro instances

To our knowledge, only three groups of authors have published approximations of the efficient set of the bTSP: Paquete *et al.* with the PLS [184] method, Paquete and Stützle with the PD-TPLS [187, 189] method (see section 5.2 in chapter 5) and Jaszkiewicz with the MOGLS method [118] (see section 3.4.3 in chapter 3). But Paquete and Stützle have shown that PD-TPLS gives better results than MOGLS [187, 189]. Jaszkiewicz and Zielniewicz have improved the results of MOGLS by using the Pareto memetic algorithm (PMA) [121] (see section 3.4.3 in chapter 3), but they have only published for these results, the running time and the R indicator. Therefore, the comparison with PMA will only be made with these two informations.

The instances used by Paquete *et al.* to test PLS are the Kro instances of 100 cities. The instances used by Paquete and Stützle to test PD-TPLS and by Jaszkiewicz and Zielniewicz to test PMA are the Kro instances with 100, 150 or 200 cities.

We first compare the results of 2PPLS, on the Kro instances, with PD-TPLS and with the PLS-3opt method of Paquete *et al.* [184], which is a Pareto local search method version 2, starting from a randomly generated solution and using the three-edge exchange neighborhood. Both algorithms have been run 50 times by the authors.

For 2PPLS, we choose the version 1 of PLS which has various advantages: not being dependent on the order in which the solutions of the population are examined and to give better results than the version 2. For the single-objective solver, we use the LKC solver.

We can see the results in Table 8.5.

Table 8.5: Comparison of 2PPLS with state-of-the-art algorithms on Kro instances.

Instance	Algorithm	$\mathcal{H}(10^8)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
KroAB100	2PPLS	**226.1059**	**1.006617**	**0.935259**	**0.280**	**2.892**	2541.70	35.57
	PLS-3opt	225.6561	1.028526	0.934996	0.549	22.096	2735.70	3000.00
	PD-TPLS	225.9605	1.017594	0.935103	0.524	9.332	896.54	**14.14**
	PD-TPLS Best	226.0509	1.010028	0.935209	0.377	3.308	1015.12	279.88
KroAC100	2PPLS	**226.3203**	**1.004604**	**0.932513**	**0.270**	**2.625**	1960.70	30.3
	PLS-3opt	225.8260	1.026469	0.932251	0.555	25.579	2133.64	3000.00
	PD-TPLS	226.1639	1.017585	0.932329	0.518	6.706	850.42	**13.72**
KroAD100	2PPLS	**227.4090**	**1.006104**	**0.934623**	**0.283**	**5.251**	1788.10	26.44
	PLS-3opt	227.1056	1.014947	0.934354	0.565	15.977	2096.06	3000.00
	PD-TPLS	227.2583	1.019599	0.934436	0.543	9.442	772.48	**13.69**
KroBC100	2PPLS	**227.3752**	**1.003957**	**0.936215**	**0.203**	**5.280**	2103.20	35.82
	PLS-3opt	227.0030	1.017446	0.935975	0.447	10.134	2403.08	3000.00
	PD-TPLS	227.2658	1.013191	0.936085	0.413	6.016	911.64	**14.65**
KroBD100	2PPLS	**226.1216**	**1.004826**	**0.934800**	**0.231**	**5.588**	1956.20	37.18
	PLS-3opt	225.5277	1.027412	0.934419	0.647	19.244	2372.96	3000.00
	PD-TPLS	226.0098	1.013345	0.934663	0.429	6.763	879.20	**14.86**
KroCD100	2PPLS	**230.8862**	**1.006435**	**0.939158**	**0.275**	**3.472**	1633.25	27.56
	PLS-3opt	230.3051	1.025861	0.938809	0.691	39.410	1846.08	3000.00
	PD-TPLS	230.7635	1.018205	0.939006	0.494	9.078	809.06	**14.02**
KroAB150	2PPLS	**592.5057**	**1.003442**	**0.942127**	**0.156**	**2.441**	3884.45	**91.21**
	PD-TPLS	592.2653	1.017489	0.942013	0.387	10.095	1558.62	116.87
KroAB200	2PPLS	**1076.0781**	**1.003257**	**0.945067**	**0.115**	**4.410**	6736.50	**211.76**
	PD-TPLS	1075.7823	1.012449	0.944988	0.293	7.229	2345.22	477.89

We remark that on five indicators, $\mathcal{H}$, $I_{\epsilon 1}$, R, D_1 and D_2, 2PPLS finds better results than PD-TPLS and PLS-3opt. The PLS-3opt method finds more potentially efficient solutions, but of worse quality than the potentially efficient solutions generated by 2PPLS. Concerning the running time, the comparison is biased because the PD-TPLS and the PLS-3opt methods have been run on a Dual Athlon with 1.2 GHz. But we can see that the running time of PLS-3opt is much higher than the running time of 2PPLS. On the other hand, the running time of PD-TPLS is lower than the running time of 2PPLS. However, Paquete and Stützle have tried to improve the results of PD-TPLS by increasing the number of aggregations used in their method, and consequently the running time. They only do that for the KroAB100 instance, and we can see the results under the name "PD-TPLS Best". We notice that although the results of PD-TPLS are improved, they remain lower than 2PPLS, and with a relative higher running time.

We now compare the results of 2PPLS to PMA, only for the R indicator. We can see the results in Table 8.6. We remark that the R indicator of PMA is better on five of the six instances of 100 cities, but worse on the 150 and 200 cities instances. On the other hand, the running time of PMA is higher (on a Pentium with 2.8 Ghz). Therefore, a future objective will be to improve the results of 2PPLS . We will see in section 8.5 a simple way to do that, even if at the end of 2PPLS we are blocked in a minimal critical non-dominated Pareto local optimum set with respect to the two-edge exchange neighborhood.

Comparison with PD-TPLS on the other instances

As state-of-the-art results are not known for all instances used in this work, we have implemented ourselves the PD-TPLS method of Paquete and Stützle [187], a method presenting few parameters contrary to many other MOGAs. This allows to produce a comparison as fair as possible with PD-TPLS, since 2PPLS and PD-TPLS are run on the same computer and share

Table 8.6: Comparison of 2PPLS with PMA on Kro instances.

Instance	Algorithm	R	Time(s)
KroAB100	2PPLS	0.935259	**36**
	PMA	**0.935280**	67
KroAC100	2PPLS	0.932513	**30**
	PMA	**0.932521**	67
KroAD100	2PPLS	**0.934623**	**26**
	PMA	0.934617	69
KroBC100	2PPLS	0.936215	**36**
	PMA	**0.936221**	73
KroBD100	2PPLS	0.934800	**37**
	PMA	**0.934809**	74
KroCD100	2PPLS	0.939158	**28**
	PMA	**0.939166**	73
KroAB150	2PPLS	**0.942127**	**91**
	PMA	0.942081	223
KroAB200	2PPLS	**0.945067**	**212**
	PMA	0.943312	567

the same data structures. However, our implementation of PD-TPLS is presumable not as good as the original implementation and the running time has to be seen as an indicator but not as an objective comparison factor. For this implementation, we have fixed the parameters of this method as follows, as done in [187]:

- The number of iterations for the number of perturbation steps of the iterated local search method used in PD-TPLS is equal to the number of cities N minus 50.
- The number of aggregations is equal to N.

We can see the results in Table 8.14. We remark that 2PPLS finds better results than PD-TPLS on all indicators, for all instances, except for the ClusteredAB300 and ClusteredAB500 instances. But the running time of 2PPLS is higher, except for the RdAB500 instance. For instances of 500 cities, the running time of 2PPLS can be very high on the Euclidean (6309 s) and clustered instances (13495 s). On the other hand, the running time of PD-TPLS remains reasonable, and does not seem dependent on the instance type.

8.4.2 Mann-Whitney test

The results of the Mann-Whitney statistical test (see section 3.5.4 in chapter 3) comparing the indicators of 2PPLS with those of PD-TPLS are presented in Table 8.8. The starting level of risk of the test has been fixed to 1%.

The results show that we can affirm with a very low risk that, for the indicators considered in this work, 2PPLS gives better or equal results than PD-TPLS on all the instances experimented in this work, except for the ClusteredAB300 instance (on $\mathcal{H}$, $I_{\epsilon 1}$ and D_1) and for the ClusteredAB500 instance (on $I_{\epsilon 1}$).

8.4.3 Outperformance relations

We now compare in term of outperformance relations the solutions obtained with 2PPLS and PD-TPLS (see section 3.5.5 in chapter 3).

We show in Figure 8.6 the results of the comparison between the potentially non-dominated points obtained with 2PPLS and with PD-TPLS, for the 300 cities instances.

Table 8.7: Comparison of 2PPLS with PD-TPLS on the other instances.

Instance	Algorithm	$\mathcal{H}(10^8)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
EuclAB100	2PPLS	**217.6296**	**1.006386**	**0.930092**	**0.325**	**4.938**	1303.70	28.60
	PD-TPLS	217.5111	1.013736	0.929976	0.521	8.063	744.70	**11.57**
EuclAB300	2PPLS	**2309.6866**	**1.124524**	**0.941704**	**16.865**	**21.612**	14749.60	783.14
	PD-TPLS	2308.9657	1.124950	0.941598	17.020	21.921	4415.40	**255.13**
EuclAB500	2PPLS	**7165.3936**	1.128923	**0.948997**	**17.244**	**21.616**	34861.95	6309.10
	PD-TPLS	7163.8715	**1.128899**	0.948927	17.395	21.829	9306.75	**1056.43**
RdAB100	2PPLS	**529.8228**	**1.021462**	**0.953365**	**0.810**	**14.170**	515.75	26.34
	PD-TPLS	528.5371	1.057139	0.952908	1.458	27.297	372.00	**12.34**
RdAB300	2PPLS	**4804.5809**	**1.728526**	**0.966936**	**20.902**	**36.295**	2006.55	351.27
	PD-TPLS	4790.3495	1.893817	0.966152	22.848	60.815	1238.70	**305.41**
RdAB500	2PPLS	**13988.0537**	**1.790178**	**0.973703**	**19.494**	**54.692**	3492.85	**1095.22**
	PD-TPLS	13951.0061	2.057356	0.972940	21.954	95.618	2081.55	1324.55
MixedAB100	2PPLS	**331.9920**	**1.015601**	**0.936814**	**0.529**	**5.768**	882.55	28.28
	PD-TPLS	331.6327	1.034143	0.936690	0.753	12.094	536.75	**11.89**
MixedAB300	2PPLS	**3410.4396**	**1.778754**	**0.956496**	**20.132**	**28.411**	5633.70	447.06
	PD-TPLS	3406.1158	1.943372	0.956254	20.722	37.980	2299.80	**288.04**
MixedAB500	2PPLS	**10440.4057**	**1.710616**	**0.960654**	**22.856**	**31.708**	13636.70	2682.45
	PD-TPLS	10429.1145	1.907713	0.960452	23.517	35.546	4816.00	**1303.07**
ClusteredAB100	2PPLS	**267.3018**	**1.007167**	**0.950017**	**0.229**	**4.937**	2490.30	55.68
	PD-TPLS	267.1498	1.019662	0.949873	0.534	8.792	848.10	**13.17**
ClusteredAB300	2PPLS	2565.4582	1.242723	**0.956624**	17.751	22.229	16885.00	1554.69
	PD-TPLS	**2566.3779**	**1.232286**	0.956616	**17.702**	**21.917**	4540.15	**293.72**
ClusteredAB500	2PPLS	8516.6889	1.183202	**0.962417**	**16.959**	24.443	42743.25	13495.33
	PD-TPLS	**8516.7115**	**1.181800**	0.962394	16.974	**23.230**	9678.15	**1239.56**

Table 8.8: Results of the Mann-Whitney test for the $\mathcal{H}$, $I_{\epsilon 1}$, R, D_1 and D_2 indicators. Comparison between 2PPLS and PD-TPLS.

Instance	$\mathcal{H}$	$I_{\epsilon 1}$	R	D_1	D_2
KroAB100	>	>	>	>	>
KroAB150	>	>	>	>	>
KroAB200	>	>	>	>	>
KroAC100	>	>	>	>	>
KroAD100	>	>	>	>	>
KroBC100	>	>	>	>	>
KroBD100	>	>	>	>	>
KroCD100	>	>	>	>	>
EuclAB100	>	>	>	>	>
EuclAB300	>	>	>	>	>
EuclAB500	>	=	>	>	>
RdAB100	>	>	>	>	>
RdAB300	>	>	>	>	>
RdAB500	>	>	>	>	>
MixedAB100	>	>	>	>	>
MixedAB300	>	>	>	>	>
MixedAB500	>	>	>	>	>
ClusteredAB100	>	>	>	>	>
ClusteredAB300	<	<	=	<	=
ClusteredAB500	=	<	>	>	=

We can see thanks to these box-plot graphs, that many points obtained with 2PPLS dominate points of PD-TPLS, for the EuclAB300, RdAB300 and MixedAB300 instances. However, on the ClusteredAB300 instance, there are more points of 2PPLS that are dominated than points that dominate points of PD-TPLS, but the difference remains low.

8.5 Data perturbation

As shown in section 8.3.1, a better quality initial population gives better results for 2PPLS and reduces the running time of the second phase.

Therefore, we present here a simple technique to improve the results of 2PPLS, that should be applied after the dichotomic method, and before the application of PLS.

As the initial population already contains many supported efficient solutions, the improvement of this population can mainly be done by generating potentially non-supported efficient solutions.

To do that, we use the "Data Perturbation" (DP) technique, originally proposed by Codenotti *et al.* for the single-objective TSP [30]. Instead of modifying the starting solution (as carried out, for instance, in the iterated local search method [157]), DP suggests to modify input data.

The functioning is as follows: we start the `Data perturbation` algorithm (see Algorithm 18) from an approximation $\widetilde{X}_E$ of the efficient set. Here the initial set is given by the first phase of 2PPLS. The input parameters of the algorithm 18 are the number K of iterations, three parameters that determine the perturbation scheme (the fixed scale *SF*, the varying scale *SV* and the local perturbation *LP*) and the cost matrices C_1 and C_2 of the bTSP. The set $\widetilde{X}_E$ is at the same time an input and an output parameter.

During an iteration k, we first compute a weight set λ by following a linear scheme; K uniformly distributed weight sets are thus generated. We then create a new cost matrix C_λ. If we solve this single-objective problem, the chances that the optimal solution of this problem is a solution already generated during the first phase are high. Therefore, we slightly perturb

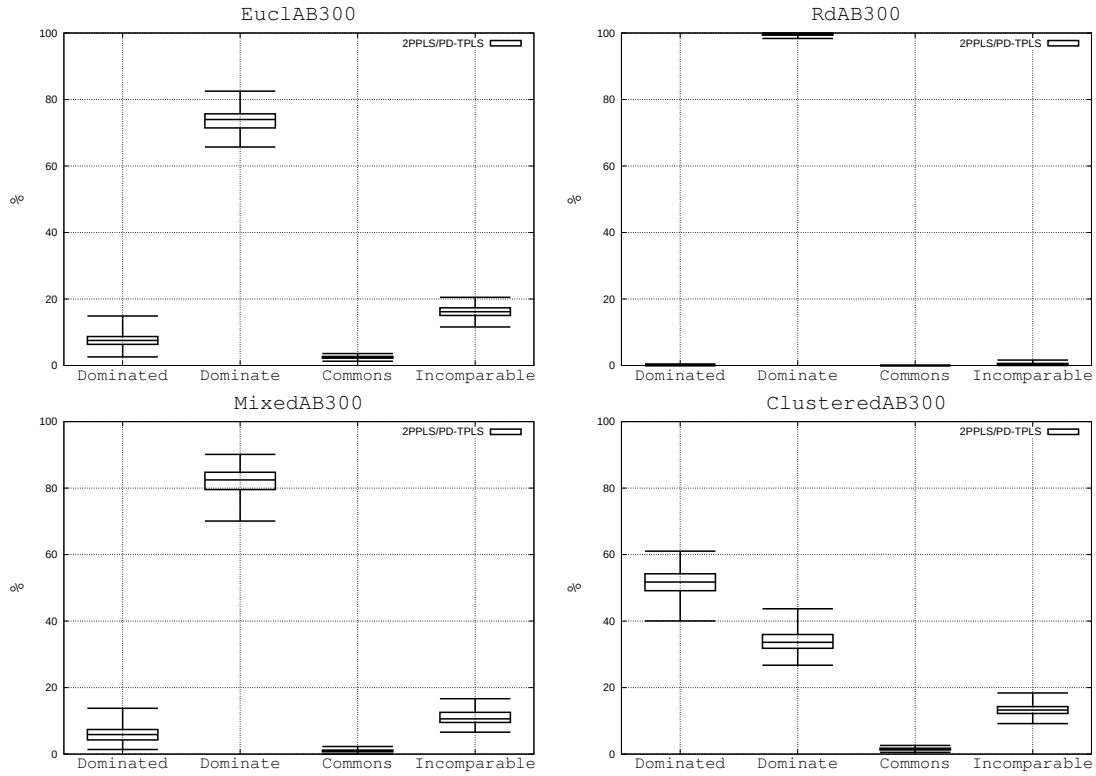

Figure 8.6: Comparison between the potentially non-dominated points obtained with 2PPLS and with PD-TPLS.

each cost $c_\lambda(i,j)$ of the matrix C_λ to find new potentially efficient solutions. To generate a perturbed instance of the bTSP, we proceed in the following way. We first fix a scale variable, called *SC*. This variable makes it possible to modulate the perturbations. A large value for *SC* gives a slightly perturbed instance while a low value gives greater perturbations. The value of *SC* is equal to a fixed scale parameter *SF* plus a value randomly generated between 0 and *SV*, where *SV* is the varying scale parameter. Then, we add or withdraw from each cost $c_\lambda(i,j)$ a small fraction $\frac{\epsilon}{SC}$ of $c_\lambda(i,j)$, for thus obtaining the perturbed cost matrix $C_{\lambda P}$. The value of ϵ is randomly generated between 0 and the local perturbation parameter *LP*. A single-objective solver, called `SolveTSP`, is then applied by considering the perturbed cost matrix $C_{\lambda P}$. We finally update the set $\widetilde{X}_E$ with the procedure `AddSolution` (see section 3.4 in chapter 3), by considering the original cost matrices C.

Hence, by applying a single-objective solver with perturbed data, new solutions, essentially potentially non-supported efficient, could be found since the data given by the linear aggregation are perturbed.

We note by 2PPLS+P the 2PPLS method plus the perturbation algorithm. The perturbation algorithm is applied after the first phase and before the PLS method.

We have represented in Figure 8.7 the evolution of the D_1 indicator and $|PE|$ of 2PPLS+P according to the running time for the KroAB100 and KroAB200 instances (the running time is controlled by the number of iterations K varying from 0 to 5000 and therefore, each point in the figure corresponds to a certain value of K). We use as solver for the first phase and for the perturbations, the LKC solver. We choose the version 1 of PLS. The parameters of the perturbation algorithm have experimentally been fixed to 1000 for *SF*, 2000 for *SV* and 100 for *LP*.

We remark that the increase of the number K of iterations gives an important improvement of the indicators and allows to reach excellent results, since the number $|PE|$ of potentially efficient solutions is increasing while the distance D_1 is decreasing. We can also see on this figure, the

Algorithm 18 `Data perturbation`

Parameters ↓: Number of iterations K, the perturbation parameters LP, SF and SV, the cost matrices C_1 and C_2 of the bTSP.
Parameters ↕: An approximation $\widetilde{X}_E$ of the efficient set.

for k=1 to K **do**
  $\lambda_1 = 1 - \frac{(k-1)}{(K-1)}$
  $\lambda_2 = 1 - \lambda_1$
  --| Creation of a new cost matrix C_λ
  $C_\lambda = \left[\lambda_1 c_1(i,j) + \lambda_2 c_2(i,j)\right]$
  --| Perturbation of the cost matrix C_λ
  $SC \leftarrow SF + \text{random}(SV)$
  for $i = 1$ to $n-1$ **do**
    for $j = i+1$ to n **do**
      $\epsilon \leftarrow \text{random}(LP)$
      $c_{\lambda P}(i,j) = \text{round}\big(c_\lambda(i,j) \pm c_\lambda(i,j) \cdot \frac{\epsilon}{SC}\big)$
  --| Resolution of the single-objective TSP by considering the cost matrix $C_{\lambda P}$
  `SolveTSP`($C_{\lambda P}$ ↓, x^t ↑)
  `AddSolution`($\widetilde{X}_E$ ↕, x^t ↓, $f(x^t)$ ↓)

effect of the increase of the running time of PD-TPLS by increasing the number of aggregation function (from n to 5000). It is noticed, that in spite of this increase, the performances of PD-TPLS stagnate and never reach the performances of 2PPLS+P.

The interest of using the perturbation algorithm after the first phase of 2PPLS seems thus important, since thanks to these perturbations, new solutions are constantly found and the quality of the approximations is increasing. However, even if the running time of the second phase of 2PPLS decreases with K since the initial population is of better quality, the total running time of 2PPLS increases with the value of K.

We finally compare the results obtained by 2PPLS+P with $K = 200$ to the results obtained by PMA for the "Kro" instances of 100 cities for which PMA obtained a better R indicator than 2PPLS. We have represented the results in Table 8.9, and we remark that now, the 2PPLS+P method manages to obtain a slightly better value for the R indicator, in similar running times.

Table 8.9: Comparison of 2PPLS+P with PMA.

Instance	Algorithm	R	Time(s)
KroAB100	2PPLS+P	**0.935294**	**62**
	PMA	0.935280	67
KroAC100	2PPLS+P	**0.932532**	**55**
	PMA	0.932521	67
KroBC100	2PPLS+P	**0.936225**	**64**
	PMA	0.936221	73
KroBD100	2PPLS+P	**0.934818**	**68**
	PMA	0.934809	74
KroCD100	2PPLS+P	**0.939182**	**52**
	PMA	0.939166	73

A weak point of the data perturbation technique is the introduction of new parameters that have to be tuned.

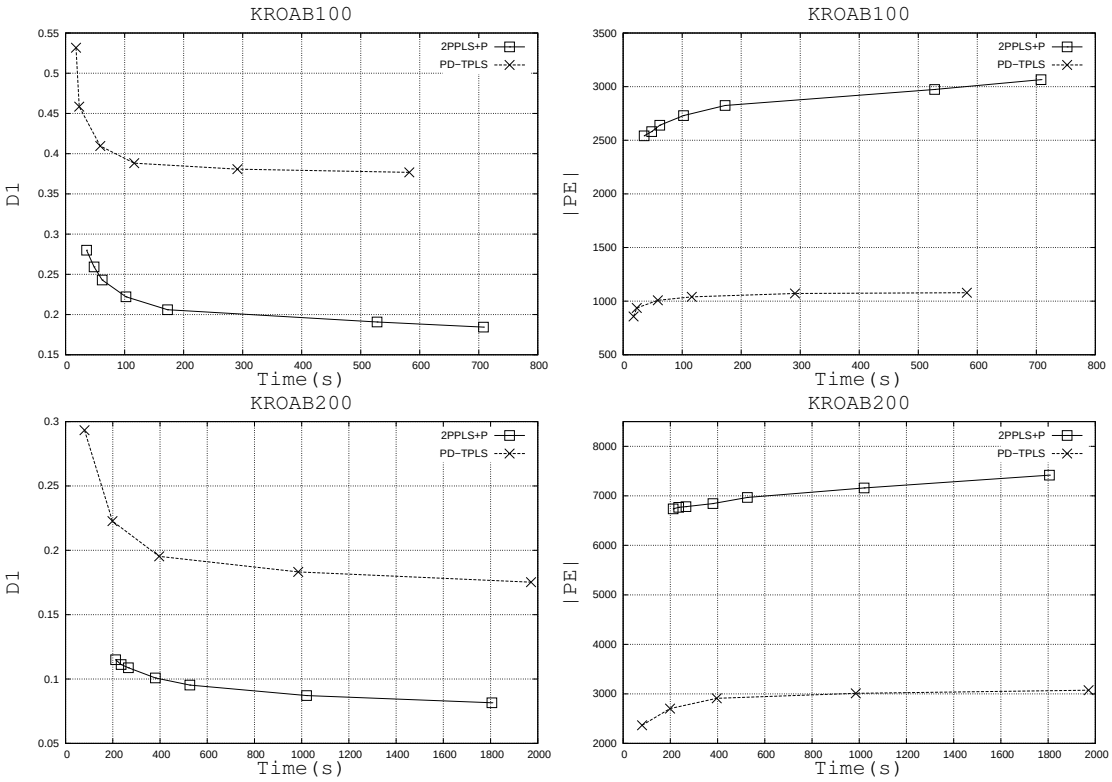

Figure 8.7: Influence of the number iterations K of the perturbation algorithm.

8.6 Speed-up techniques

8.6.1 Introduction

We have seen in the preceding sections that 2PPLS obtained better results than state-of-the-art algorithms, but however, the running time of 2PPLS was significantly higher than PD-TPLS. We have introduced the data perturbation technique in order to improve the results of 2PPLS, but the running time is still increasing when this technique is applied.

We have thus adapted some speed-up techniques [20, 123], traditionally used in single-objective optimization, to the biobjective case. The speed-up techniques are integrated into 2PPLS.

We have represented in Figure 8.8 the evolution of the running time of the two phases (P1 and PLS) of 2PPLS according to the instance size. We use the instances KroAB whose the size varies from 100 to 1000. In this section, we will always use the LKC solver in phase 1 as well as the first version of PLS.

We remark that the running time of the first phase increases more or less linearly according to the instance size. On the other hand, the running time of the second phase, PLS, strongly increases (6000s for the instance of 500 cities). Indeed, in the second phase, we totally explore the neighborhood of every solution of a population, by making two-edge exchange moves. Since for each solution of the population the number of neighbors generated is equal to $\frac{n(n-3)}{2}$, it takes $\mathcal{O}(n^2)$ time to generate neighbors from one solution of the population.

Therefore, solving instances of more than 500 cities with 2PPLS without speed-up techniques is practically impossible. Effectively, we did not manage to solve the instances of 750 and 1000 cites in a reasonable time with 2PPLS. Therefore, speed-up techniques will be useful to reduce the running time of 2PPLS, while keeping better quality results than state-of-the-art methods. Many speed-up techniques have been developed for the single-objective TSP [20, 123], but

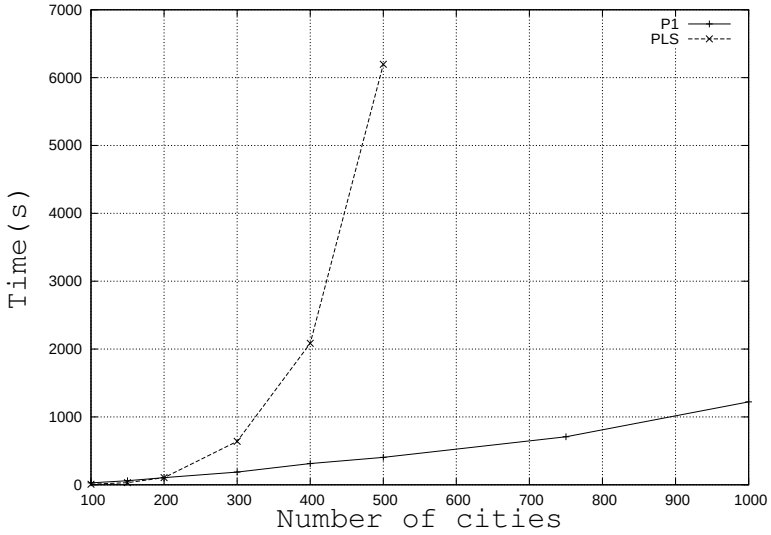

Figure 8.8: Evolution of the running time of the two phases.

to our knowledge, none of these techniques has been adapted to the resolution of the bTSP (excluding biobjective instances resolved by a method using aggregation functions to transform the biobjective problem into several single-objective problems).

We present thus in the next section speed-up techniques for solving the bTSP with 2PPLS, in order to reduce the running time of the second phase.

Before presenting the different speed-up techniques, let's take a look at the edges used by the solutions of an efficient set. As for biobjective instances, the edges can be represented in a two-dimensional graph (the x-coordinate and y-coordinate of the graph are respectively given by the first cost and the second cost of the edges), we will employ such representation to study what are the edges of a biobjective instance used by the efficient set.

We have represented in Figure 8.9, above, all the 4950 edges of the biobjective instance KroAB100. Below, we have represented the edges used by a near-efficient set, which is a very good approximation of the efficient set, obtained with 2PPLS and the data perturbation technique.

It is noted that only a small proportion of the edges are used by the near-efficient set, and the edges that are bad for both costs are not used at all. This observation is important to develop appropriate speed-up techniques.

In the second figure, we also add frequencies with which the edges are used by the solutions of the near-efficient set, and we remark that well-located edges (both costs are low) are intensively employed (near to 100% for certain edges, what means that almost all solutions use these edges) while others are only rarely used. But the relation between the location of the edges and the frequencies is not clear and would be difficult to take into account.

8.6.2 Candidate list for the biobjective TSP

A classic speed-up technique for solving single-objective TSP is the candidate list. This speed-up technique is based on the observation of Steiglitz and Weiner [213]: for an improving two-edge exchange move where (t_1, t_2) and (t_3, t_4) are the leaving edges and where (t_1, t_4) and (t_2, t_3) are the entering edges (see Figure 8.10), it must be the case that:

$$\text{Either } c_1(t_1, t_2) > c_1(t_2, t_3) \text{ or } c_1(t_3, t_4) > c_1(t_1, t_4) \text{ or both} \tag{8.1}$$

(where c_1 represents the single-objective cost). That is to say, the cost of one of the entering edges must be lower than the cost of one of the leaving edges.

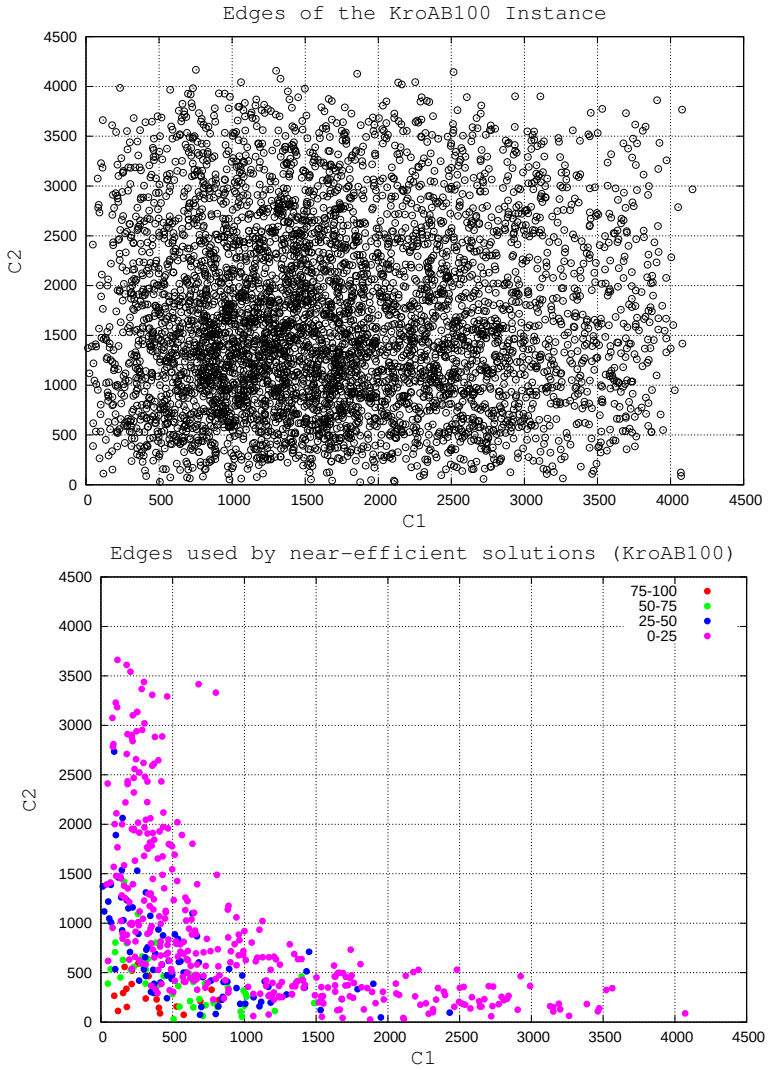

Figure 8.9: Edges of the KroAB100 instance and edges used by near-efficient solutions.

To take advantage of this observation, a first step is to compute for each city t_i a static list containing the cities in order of increasing cost with t_i (the candidate list). The size of the list is limited to a reasonable size, compromise between quality and running time. For example,

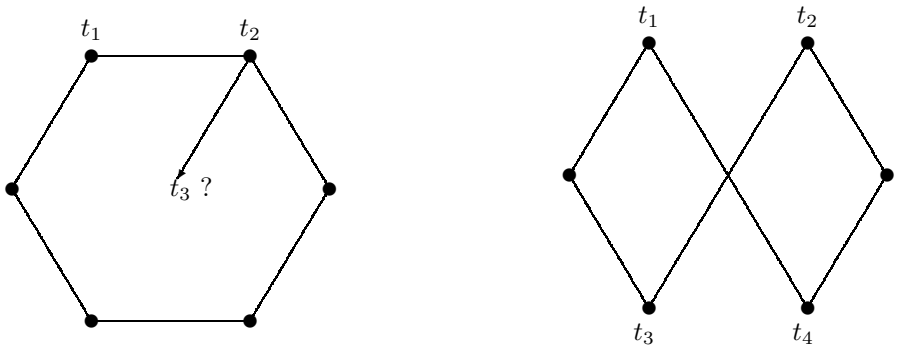

Figure 8.10: Two-edge exchange move.

in [123], Johnson and McGeoch recommend a size equal to 15.

All cities are then considered as starting cities for the two-edge exchange moves. For a starting city t_1 with t_2 the next city in the tour, to consider candidates for t_3 (see Figure 8.10), we only need to start at the beginning of the candidate list of t_2 and proceed down it until $c_1(t_2, x) > c_1(t_1, t_2)$ or when the end of the candidate list has been reached. To check all possibilities, it is also necessary to start at the beginning of the candidate list of t_1 and proceed down it until $c_1(t_1, x) > c_1(t_1, t_2)$ or when the end of the list has been reached. So, for each starting city, two candidate lists are explored: the candidate list of the starting city and the candidate list of the city following the starting city in the tour. As each city of the current tour is considered as starting city, each candidate list is explored two times.

We can then consider two different techniques: to consider the first improvement move, or among all moves, the best improvement move [102]. If the first improvement technique is used, the examination of the candidate lists is stopped as soon as an improvement move has been found; if there is one. If the best improvement technique is used, among all the improving moves, the move that allows to produce the best improvement is considered.

This technique is very efficient, and allows to considerably reduce the running time of single-objective heuristics with very small degradations of the quality of the solutions obtained [123].

For the bTSP, for an improving two-edge exchange move where (t_1, t_2) and (t_3, t_4) are the leaving edges and where (t_1, t_4) and (t_2, t_3) are the entering edges, it must be the case that:

$$\text{Either } c(t_1, t_2) \not\prec c(t_2, t_3) \text{ or } c(t_3, t_4) \not\prec c(t_1, t_4) \text{ or both} \tag{8.2}$$

(where c represents the cost vector, of dimension two in the biobjective case). That is to say, one of the entering edges must not to be dominated by one of the leaving edges. Otherwise, the move will not lead to a new non-dominated tour as the cost vector of the entering edges (equal to $c(t_2, t_3) + c(t_1, t_4)$) will be dominated by the cost vector of the leaving edges (equal to $c(t_1, t_2) + c(t_3, t_4)$).

In the biobjective case, it is no more possible to sort each candidate list, since there is no more total order between the cost vectors c. We have thus to explore each candidate list of each city until the end of the list has been reached. We see that this technique will not be as effective as in the single-objective case since each candidate list has to be explore until the end. For this reason, we do not take into account the relation (8.2) in the exploration of the candidate lists. Therefore, to check all possibilities, each candidate list has to be explored only one time.

We present below how to create the candidate lists in the biobjective case. Three different techniques are presented.

k-nearest neighbors

A simple way to consider both costs is to create two lists for each city. The first list contains the cities in order of increasing cost, for the first objective, and the second, the cities in order of increasing cost, for the second objective. We then merge the two lists to obtain a unique candidate list of size $2 * k$ by paying attention to two things: the same city cannot appear twice in the same list (as the same city can appear in both lists) and if the city t_j is in the candidate list of the city t_i, the city t_i cannot be in the candidate list of the city t_j, to avoid double evaluation of the same move.

We have represented in Figure 8.11 the edges that are taken into account for the two-edge exchange moves (called candidate edges) for $k = 1$ and $k = 5$. We see that with this technique edges having high costs for both objectives are not candidates.

Data dominance relations

Another way to create candidate lists is to use data dominance relations. Indeed, as we have seen in Figure 8.9, edges that are dominated by many other edges do not appear in the near-efficient set.

Therefore, to determine the edges that will be used, we associate to each edge a rank, based on the dominance ranking developed by Goldberg [91] (see section 3.4.2 in chapter 3). As a

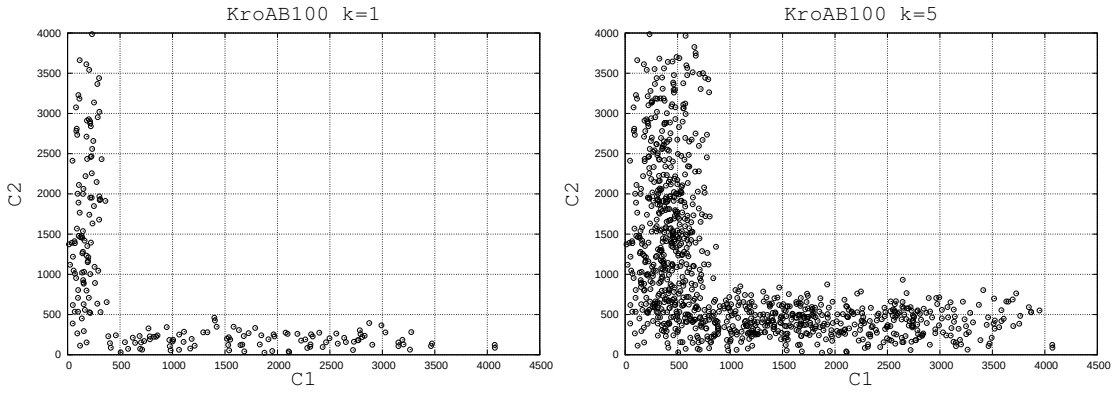

Figure 8.11: Illustration of k-nearest neighbors technique for the creation of the candidate edges.

reminder, this ranking works as follows. All non-dominated edges have a rank equal to 0. These edges are then removed, and the following non-dominated edges obtain a rank equal to 1. This process is repeated until a rank equal to the value given by a parameter D has been obtained.

We show in Figure 8.12 the representation of the edges, for $D = 0$, $D = 1$, $D = 10$ and $D = 20$. We remark that with this technique, the set of candidate edges better fits, visually, to the edges used by the near-efficient set (Figure 8.9), than with the k-nearest neighbors technique (Figure 8.11).

At the end, we create for each city a candidate list by only considering the candidate edges given by the data dominance relations. To do that, we explore the set of candidate edges, and for each candidate edge (t_i, t_j), we add the city t_j to the candidate list of the city t_i (if t_i is not already in the candidate list of t_j).

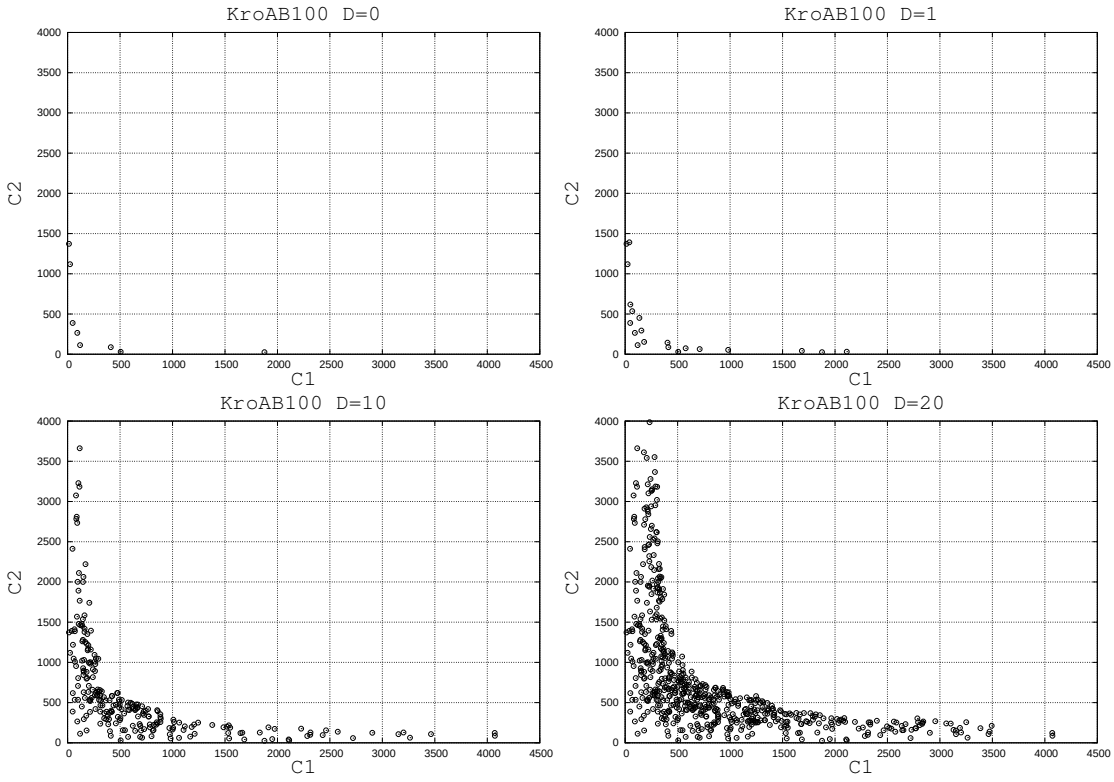

Figure 8.12: Illustration of the data dominance relations for the creation of the candidate edges.

Use of the edges used by the solutions of the first phase

As fixing a good value for k or D is not necessarily obvious, we propose here a simple way to determine candidate edges, without having to fix a value for k or D.

The idea comes from the following observation: after application of the first phase of 2PPLS, a set of potentially efficient solutions is already discovered and it would be relevant to retrieve information from this set. Gandibleux *et al.* [78] have also exploited this idea in the case of scheduling and knapsack problems.

We have represented in Figure 8.13 all the edges used in at least one solution generated during the first phase and edges used by the near-efficient set of the KroAB100 instance. We can see that both sets of candidate edges are very close. It thus seems that using only the edges found after the first phase for the creation of the candidate lists will already give good results.

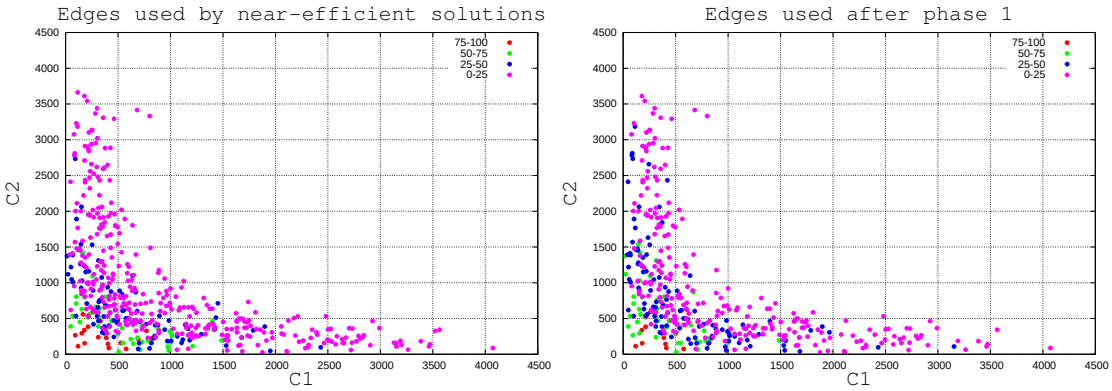

Figure 8.13: Edges used by a near-efficient set and by the set obtained after the first phase (KroAB100 instance).

To create the candidate lists, we explore the set of candidate edges obtained during the first phase, and for each candidate edge $\{t_i, t_j\}$, we add the city t_j to the candidate list of the city t_i, if t_i is not already in the candidate list of t_j (to avoid to realize the same move twice).

We will call this speed-up technique "SpeedP1".

8.6.3 Don't look bits

In the previous speed-up techniques, all cities are considered as starting cities for the two-edge exchange moves. It is however possible to not consider all cities, with a simple heuristic rule, often implemented in single-objective heuristics. This rule is known under the name "don't look bits" [20]. Here the observation is that if a starting city t_1 previously failed to find an improving move (a move that generates a new potentially non-dominated tour), and if the neighbors of t_1 in the tour have not changed since that time, the probability that an improved move will be found starting from t_1 is low.

We exploit this observation by means of special bit for each city. Before applying the PLS method, we associate to each solution of the population a boolean array don't look bits, containing bits all turned off. For a solution, the bit for city c is turned on whenever a search for an improving move with $t_1 = c$ fails and is turned off whenever its predecessors and successors have changed, that is to say an improving move is performed in which c is an endpoint of one of the deleted edges.

When we try to generate non-dominated tours from a solution of the population, we ignore all starting cities t_1 whose bits given by the array don't look bits of the solution are switched on.

8.6.4 Analysis of the results obtained by the different speed-up techniques

We first analyze in this section the results obtained by the speed-up techniques, based on different figures showing the evolution of the D_1 and R indicators according to the running time of the second phase of 2PPLS (the speed-up techniques have no influence on the running time of the first phase). The average of the indicators is made over only three runs. In this case, this number of runs is enough since the aim of the different figures is only to see which speed-up techniques dominate the others for different lengths of the candidate lists. Moreover, the stochasticity of 2PPLS comes only from the first phase.

Comparison between k-nearest and data dominance relations

We compare the two following techniques to create the candidate lists: candidate lists based on k-nearest neighbors and candidate lists based on data dominance relations.

To do that, we vary the value of k for the k-nearest technique (from 2 to 30) and the value of D for the data dominance relations technique (from 0 to 60 for the KroAB200 instance and from 0 to 70 for the KroAB300 instance).

We have represented in Figure 8.14 the values obtained by the k-nearest neighbor technique (under the name "KNearest") and the data dominance relations technique (under the name "Dominance") for the D_1 and R indicators according to the running time, for the KroAB200 and KroAB300 instances (the running time is controlled by the values of k or D, each point in the figures corresponds thus to a certain value of k or D).

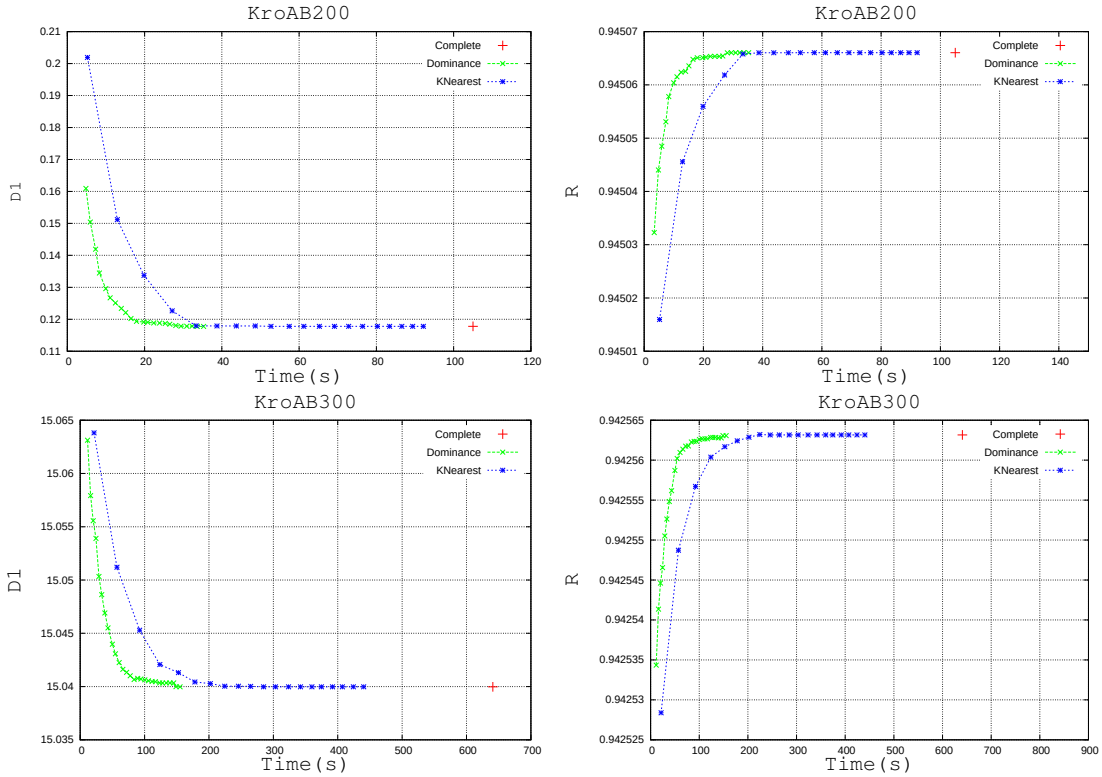

Figure 8.14: Comparison between k-nearest and data dominance relations.

We can see that the Dominance technique is more efficient than the KNearest technique: for the same running time, the values of the D_1 and R indicators obtained with Dominance are better than the values obtained with KNearest. Obviously, when a certain value for D or k is reached, the results with these two techniques are similar, since the sets of candidate edges become the same.

We have also represented the results of the complete exploration of the neighborhood (under the name "Complete") and we can see that both techniques allow to improve the running time (with a speed-up factor equal to about 3) while keeping results of same quality.

The values of D that allow to obtain the same quality result than the complete exploration are equal to 60 for the KroAB200 instance and 70 for the KroAB300 instance. For k, the values are respectively equal to 12 and 16.

Comparison between data dominance relations and speedP1

We have represented in Figure 8.15 the comparison for the D_1 and R indicators according to the running time between candidate lists created with the Dominance technique and candidate lists created from the edges used after the first phase ("SpeedP1"), for the KroAB200 and KroAB300 instances (please note that the scale of this figure is different from the scale of the preceding ones). We can see that the SpeedP1 technique allows to obtain better results than the Dominance technique: for the same running time the indicators D_1 and R found by SpeedP1 are better than the indicators found by Dominance.

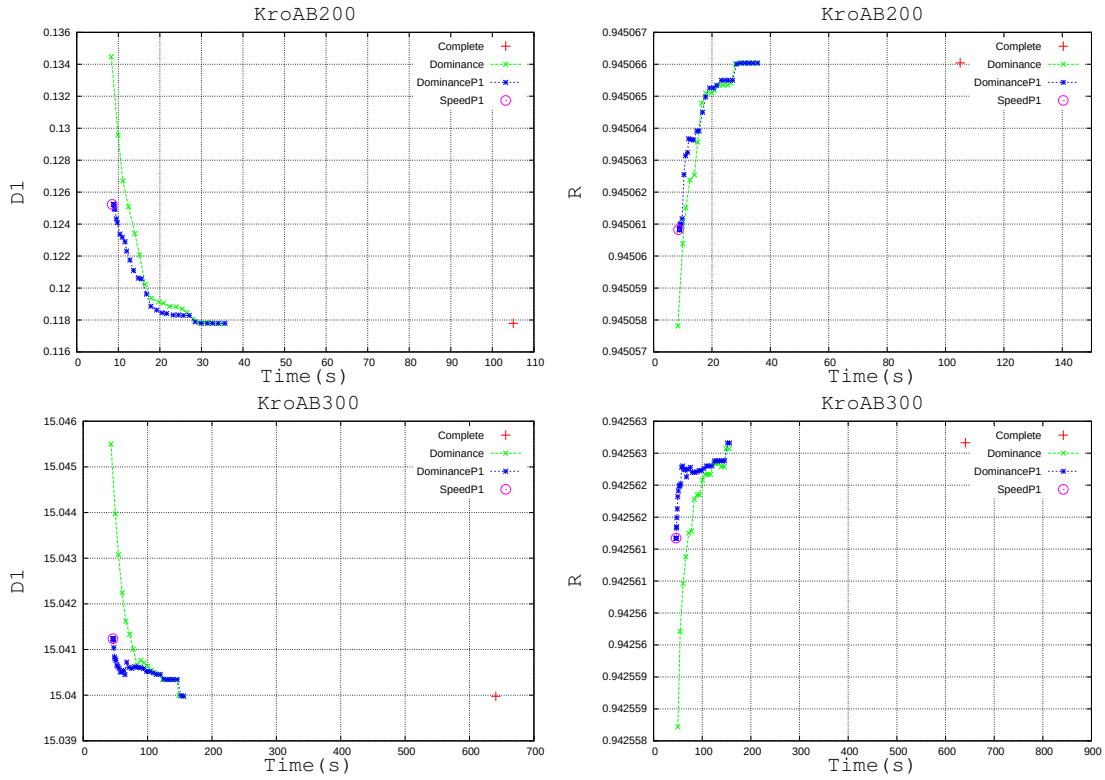

Figure 8.15: Comparison between data dominance relations and edges used by the solutions found after the first phase.

Comparing to the complete exploration of the neighborhood, the results given by SpeedP1 are very close with a much lower running time (the speed-up factor is equal to about 10). Moreover, we can improve the results given by SpeedP1 by adding edges that were not used after the first phase, in the order given by the data dominance relations. We can see these results on the same figure under the name "DominanceP1". By increasing D and therefore the running time, the results given by SpeedP1 are better and are finally as good as the complete exploration, always with a lower running time. Also, after a certain running time, the results given by DominanceP1 and Dominance are the same given that from a certain value of D, the sets of candidate edges are equal for both techniques.

Use of don't look bits

We can see in Figure 8.16 the results obtained with the data dominance relations without and with the don't look bits technique (respectively under the name "Dominance" and "Dominance Dlb").

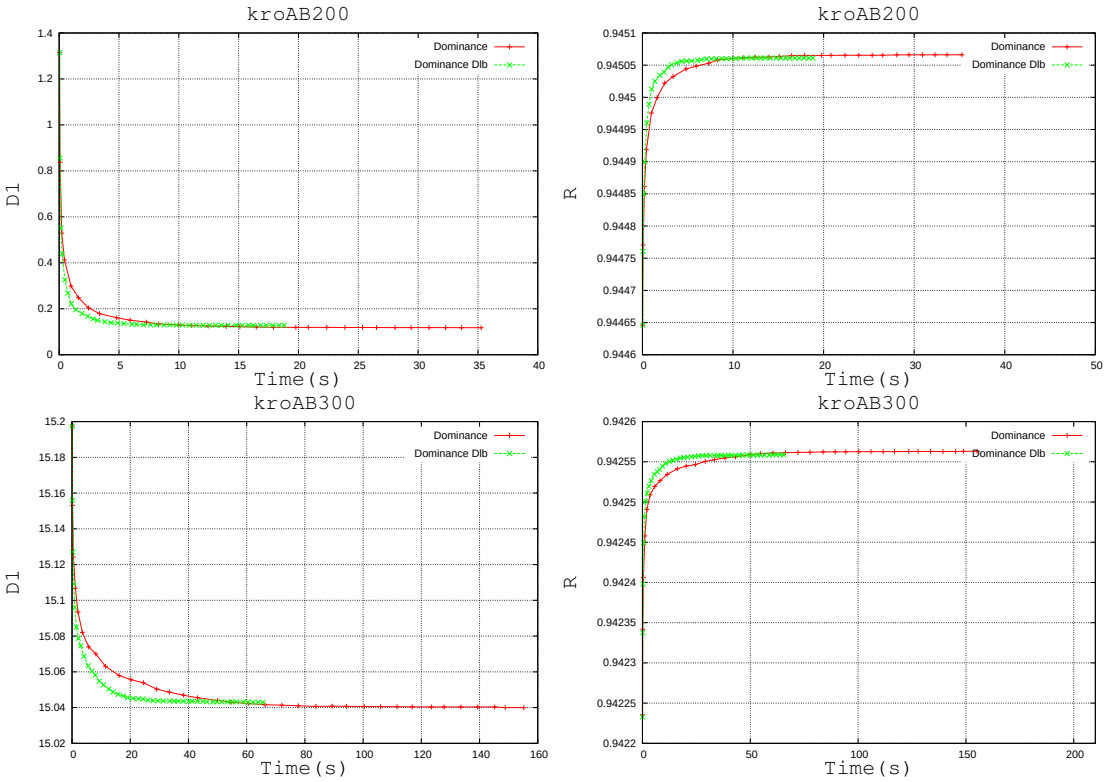

Figure 8.16: Data dominance relations and don't look bits.

We remark that for a low running time, the don't look bits technique gives better results on the D_1 and R indicators for the KroAB200 and KroAB300 instances. But by increasing D and therefore the running time, the performances without don't look bits are better. Furthermore, with the don't look bits technique, we do not manage to reach the same quality results than without, even when the value of D is increased. But the difference remains low.

We can see in Figure 8.17 the results obtained with the speedP1 technique without and with the don't look bits technique (respectively under the name "SpeedP1" and "SpeedP1 Dlb") and the results obtained by adding edges that were not used after the first phase, in the order given by the data dominance relations, without and with the don't look bits technique (respectively under the name "DominanceP1" and "DominanceP1Dlb").

We remark that the results obtained with the don't look bits technique are of worse quality than without (for a same running time, the indicators without don't look bits technique are better). But this technique remains interesting, since it allows to give approximations of relatively good quality in a low running time.

8.6.5 Comparison between the speed-up techniques

In this section, we compare, in a more accurate way, the speed-up techniques between them and also with 2PPLS without speed-up techniques. This comparison is realized for the KroAB instance of 200 cities or more. For these comparisons, we make the average of the indicators over twenty executions.

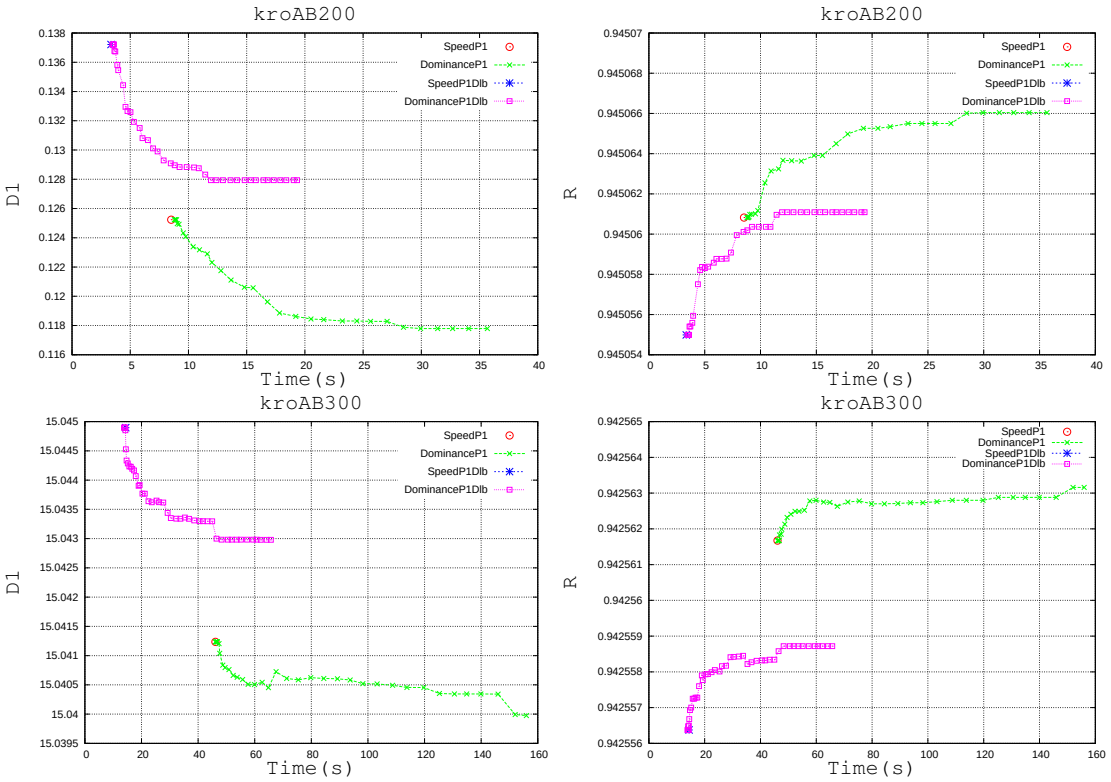

Figure 8.17: SpeedP1 and don't look bits.

Comparison based on the mean of indicators

We can see in Table 8.10 the results obtained with the different methods, that is to say:

- The standard 2PPLS method, that is the method without speed-up techniques (/);
- 2PPLS with the don't look bits technique (Dlb);
- 2PPLS with the speed-up technique based on the edges used after the first phase (SpeedP1) and
- 2PPLS with both techniques (SpeedP1+Dlb).

These results are for the KroAB200, KroAB300, KroAB400 and KroAB500 instances. The running time indicated is decomposed between the time of the first phase (not dependent on the speed-up techniques) and the time of the second phase.

We remark that the values of the indicators are practically the same and thus almost not deteriorated by the speed-up techniques. Moreover, the running time of PLS (second phase of 2PPLS) is strongly reduced by employing the speed-up techniques. We also observe that the SpeedP1 technique is more efficient than the Dlb technique.

We have also tried to solve instances of size equal to 750 and 1000. For these instances, it was not possible to apply the standard 2PPLS method while keeping reasonable running times. We have represented these results in Table 8.11. Even with the SpeedP1 technique, the running time is high (more than 7000s for the KroAB1000 instance). But by applying the don't look bits technique coupled with the SpeedP1 technique, it is possible to strongly reduce the running time (about 2000s). It should be noted that the number of potentially efficient solutions is very high: nearly equal to 100000 for the KroAB1000 instance.

Table 8.10: Comparison between speed-up techniques(1).

Instance	Speed-up	$\mathcal{H}(10^8)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
KroAB200	/	1076.0781	1.003257	0.945067	0.115	4.410	6736.50	106 + 106
	Dlb	1076.0600	1.003987	0.945062	0.125	4.410	6168.00	106 + 26
	SpeedP1	1076.0619	1.003825	0.945062	0.123	4.410	6210.75	106 + 9
	SpeedP1+Dlb	1076.0411	1.004584	0.945056	0.134	4.410	5569.95	106 + 3
KroAB300	/	2409.2880	1.141328	0.942563	15.040	18.692	14853.10	188 + 640
	Dlb	2409.2520	1.141328	0.942558	15.045	18.716	13446.85	188 + 125
	SpeedP1	2409.2686	1.141328	0.942561	15.042	19.048	14101.25	188 + 44
	SpeedP1+Dlb	2409.2324	1.141328	0.942556	15.047	19.052	12514.85	188 + 14
KroAB400	/	4306.7424	1.135376	0.944511	16.781	36.307	22068.25	312 + 2085
	Dlb	4306.6795	1.135376	0.944505	16.788	36.309	19249.45	312 + 339
	SpeedP1	4306.7155	1.135393	0.944509	16.784	36.583	21098.90	312 + 122
	SpeedP1+Dlb	4306.6486	1.135393	0.944503	16.790	36.585	18092.00	312 + 33
KroAB500	/	6819.1741	1.125555	0.946919	17.332	22.559	33690.80	404 + 6197
	Dlb	6819.0712	1.125555	0.946913	17.339	22.575	28459.75	404 + 784
	SpeedP1	6819.1438	1.125555	0.946917	17.334	22.568	32245.95	404 + 327
	SpeedP1+Dlb	6819.0360	1.125555	0.946911	17.340	22.580	26496.60	404 + 73

Table 8.11: Comparison between speed-up techniques(2).

Instance	Speed-up	$\mathcal{H}(10^8)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
KroAB750	Dlb	15960.4121	1.113490	0.952319	17.181	31.616	50162.40	708 + 3367
	SpeedP1	15960.5828	1.113525	0.952323	17.174	32.233	59345.00	708 + 1593
	SpeedP1+Dlb	15960.3403	1.113533	0.952317	17.184	31.414	46332.20	708 + 266
KroAB1000	Dlb	27922.4283	1.111325	0.954128	17.538	23.300	79793.85	1222 + 10872
	SpeedP1	27922.6908	1.111352	0.954132	17.531	23.581	97119.10	1222 + 5984
	SpeedP1+Dlb	27922.3409	1.111353	0.954127	17.540	23.382	73990.75	1222 + 798

Mann-Whitney test

The results of the Mann-Whitney statistical test comparing the indicators of 2PPLS with those of 2PPLS+SpeedP1 are presented in Table 8.12, for the KroAB200, KroAB300, KroAB400 and KroAB500 instances. The starting level of risk of the test has been fixed to 1%.

Table 8.12: Results of the Mann-Whitney test for the $\mathcal{H}$, $I_{\epsilon 1}$, R, D_1 and D_2 indicators (2PPLS+SpeedP1 compared to 2PPLS).

Instance	$\mathcal{H}$	$I_{\epsilon 1}$	R	D_1	D_2
KroAB200	$<$	$<$	$<$	$<$	$=$
KroAB300	$=$	$=$	$<$	$=$	$=$
KroAB400	$=$	$=$	$=$	$=$	$=$
KroAB500	$=$	$=$	$=$	$=$	$=$

We remark that for the KroAB200 instance, the standard 2PPLS method is statistically better than the 2PPLS+SpeedP1 method on all indicators, but D_2. For the KroAB300, 2PPLS is better only on the R indicator. For the rest of the instances, that is the large-scale instances, there is no statistically significant difference between both methods.

Outperformance relations

We now compare in term of outperformance relations the solutions obtained with 2PPLS and with 2PPLS+SpeedP1, for the 300 and 500 cities instances.

We can see in Figure 8.18 that about 80% of the solutions obtained with 2PPLS+SpeedP1 are also generated by 2PPLS. Only about 15% of the solutions of 2PPLS+SpeedP1 are dominated by 2PPLS. Moreover, 2PPLS+SpeedP1 allows to generate some new solutions that dominate solutions obtained with 2PPLS.

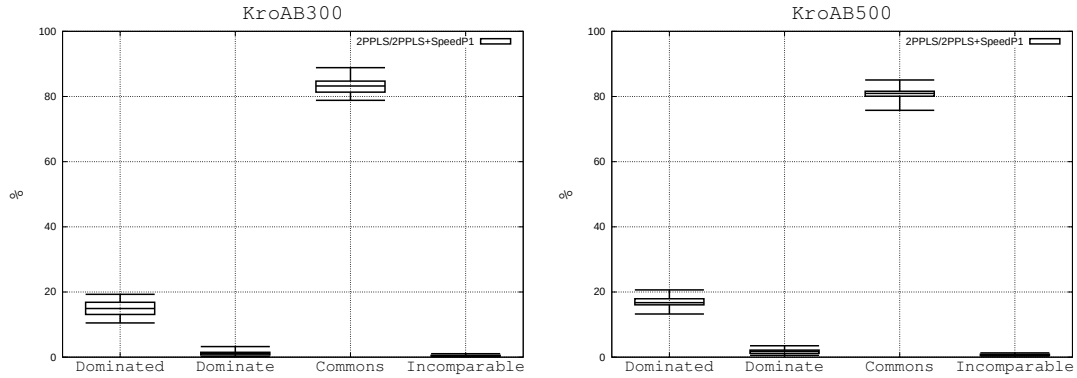

Figure 8.18: Comparison between the potentially non-dominated points obtained with 2PPLS and with 2PPLS+SpeedP1 for the KroAB300 and KroAB500 instances.

Evolution of the running time

In Figure 8.19, we have represented the evolution of the gain in running time by applying the SpeedP1 technique in comparison with the standard 2PPLS method.

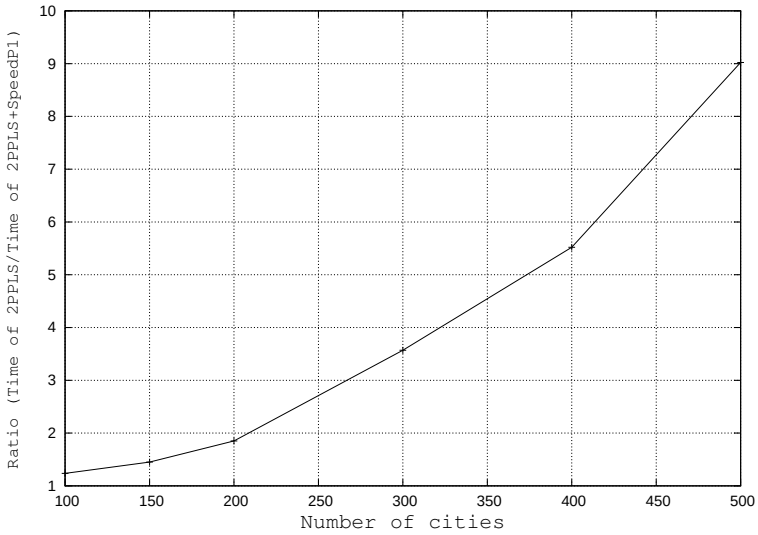

Figure 8.19: Evolution of the gain in running time between SpeedP1 and the standard 2PPLS method.

We can see than the gain in running time is strongly increasing according to the instance size.

Evolution of the number of edges and of the number of neighbors

In Figure 8.20, we show two things: the evolution of the ratio equal to the number of edges used by the solutions generated during the first phase and the total number of edges, and the ratio between the number of neighbors generated by 2PPLS+SpeedP1 and the number of neighbors generated by 2PPLS. Both ratios are expressed in percentage.

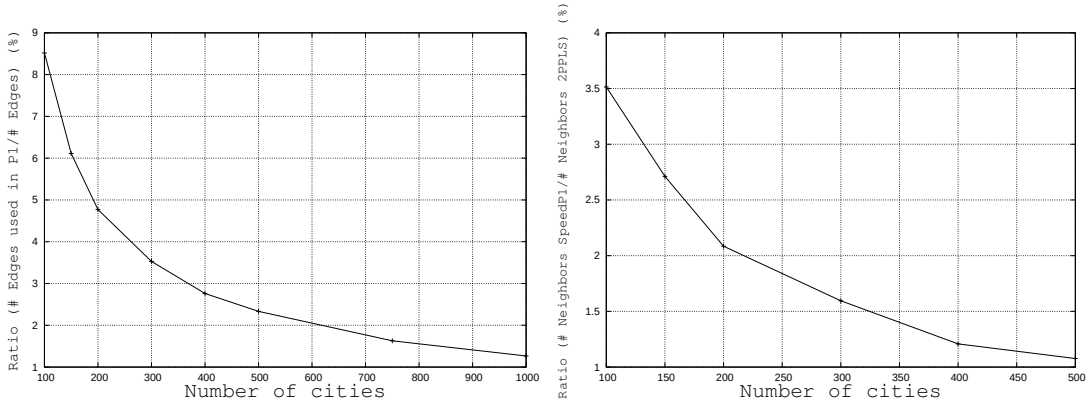

Figure 8.20: Evolution of the ratio between the number of edges used by the solutions generated during the first phase and the total number of edges (%) and evolution of the ratio between the number of neighbors generated by 2PPLS+SpeedP1 and the number of neighbors generated by 2PPLS (%).

As we can see, both ratios are decreasing, what can explain why the gain in running time is increasing according to the instance size.

8.6.6 Comparison between 2PPLS with SpeedP1 and PD-TPLS

Comparison based on the mean of the indicators

We first compare 2PPLS with SpeedP1 to PD-TPLS on the KroAB instances of 300 cities and more. The results of PD-TPLS come from our implementation with the same parameters than exposed in section 8.4.1.

We can see the results in Table 8.13 and we remark that 2PPLS with the SpeedP1 technique (2PPLS+SpeedP1) allows to obtain better results in lower running times than PD-TPLS, for the indicators considered, except for $I_{\epsilon 1}$ on the KroAB500 instance and for D_2 on the KroAB300 and KroAB750 instances.

Table 8.13: Comparison between 2PPLS+SpeedP1 and PD-TPLS on the Kro instances.

Instance	Method	$\mathcal{H}(10^8)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
KroAB300	2PPLS+SpeedP1	**2409.2686**	**1.141328**	**0.942561**	**15.042**	19.048	14101.25	**232**
	PD-TPLS	2408.5770	1.141768	0.942473	15.155	**18.903**	4464.90	258
KroAB400	2PPLS+SpeedP1	**4306.7155**	**1.135393**	**0.944509**	**16.784**	**36.583**	21098.90	**434**
	PD-TPLS	4305.6989	1.136508	0.944428	16.912	38.874	6478.75	591
KroAB500	2PPLS+SpeedP1	**6819.1438**	1.125555	**0.946917**	**17.334**	**22.568**	32245.95	**731**
	PD-TPLS	6817.7144	**1.125362**	0.946848	17.468	22.751	8634.45	1137
KroAB750	2PPLS+SpeedP1	**15960.5828**	**1.113525**	**0.952323**	**17.174**	32.233	59345.00	**2301**
	PD-TPLS	15957.9824	1.114744	0.952270	17.298	**30.861**	14511.80	3579
KroAB1000	2PPLS+SpeedP1	**27922.6908**	**1.111352**	**0.954132**	**17.531**	**23.581**	97119.10	**7206**
	PD-TPLS	27918.6072	1.113044	0.954090	17.653	24.598	21984.25	8103

The results for the other instances are given in Table 8.14. We keep the same parameters

for PD-TPLS, except for the 100 cities instances where the number of aggregations is higher than n in order to obtain comparable running times between 2PPLS+SpeedP1 and PD-TPLS. The number of aggregations is equal to 250 for the 100 cities instances, excepted for the ClusteredAB100 instance where this number has been set to 400.

Table 8.14: Comparison between 2PPLS+SpeedP1 with PD-TPLS on the other instances.

Instance	Algorithm	$\mathcal{H}(10^8)$	$I_{\epsilon 1}$	R	D_1	D_2	$\|PE\|$	Time(s)
EuclAB100	2PPLS+SpeedP1	**217.62**	**1.007141**	**0.930081**	**0.345**	5.091	1196.90	**25.53**
	PD-TPLS	217.55	1.013423	0.930009	0.460	**4.072**	811.35	29.47
EuclAB300	2PPLS+SpeedP1	**2309.67**	**1.124524**	**0.941701**	**16.868**	**21.633**	14050.90	258.56
	PD-TPLS	2308.97	1.124950	0.941598	17.020	21.921	4415.40	**255.13**
EuclAB500	2PPLS+SpeedP1	**7165.35**	1.129026	**0.948995**	**17.247**	22.072	33017.30	**692.88**
	PD-TPLS	7163.87	**1.128899**	0.948927	17.395	**21.829**	9306.75	1056.43
RdAB100	2PPLS+SpeedP1	**529.77**	**1.025300**	**0.953339**	**0.932**	**14.325**	438.30	**25.46**
	PD-TPLS	528.92	1.042416	0.953061	1.195	30.620	419.40	30.89
RdAB300	2PPLS+SpeedP1	**4804.48**	**1.728586**	**0.966925**	**20.910**	**36.424**	1766.95	**305.36**
	PD-TPLS	4790.35	1.893817	0.966152	22.848	60.815	1238.70	305.41
RdAB500	2PPLS+SpeedP1	**13987.92**	**1.790290**	**0.973697**	**19.502**	**55.098**	3127.85	**816.40**
	PD-TPLS	13951.01	2.057356	0.972940	21.954	95.618	2081.55	1324.55
MixedAB100	2PPLS+SpeedP1	**331.96**	**1.017122**	**0.936782**	**0.579**	**6.006**	793.10	**26.24**
	PD-TPLS	331.74	1.032013	0.936741	0.659	12.754	585.10	30.15
MixedAB300	2PPLS+SpeedP1	**3410.40**	**1.778747**	**0.956492**	**20.135**	**28.422**	5202.05	**238.45**
	PD-TPLS	3406.12	1.943372	0.956254	20.722	37.980	2299.80	288.04
MixedAB500	2PPLS+SpeedP1	**10440.35**	**1.710601**	**0.960652**	**22.858**	**31.865**	12925.40	**865.55**
	PD-TPLS	10429.11	1.907713	0.960452	23.517	35.546	4816.00	1303.07
ClusteredAB100	2PPLS+SpeedP1	**267.28**	**1.007686**	**0.949999**	**0.274**	10.426	2184.05	**50.02**
	PD-TPLS	267.21	1.019305	0.949887	0.442	**6.015**	989.45	53.56
ClusteredAB300	2PPLS+SpeedP1	2565.34	1.243181	**0.956617**	17.763	24.363	15511.10	366.25
	PD-TPLS	**2566.38**	**1.232286**	0.956616	**17.702**	**21.917**	4540.15	**293.72**
ClusteredAB500	2PPLS+SpeedP1	8516.35	1.183524	**0.962412**	**16.974**	28.404	40100.60	1517.68
	PD-TPLS	**8516.71**	**1.181800**	0.962394	**16.974**	**23.230**	9678.15	**1239.56**

We remark that 2PPLS+SpeedP1 finds better results than PD-TPLS on all indicators for the random instances and mixed instances. For the Euclidean instances, 2PPLS+SpeedP1 finds better results on all indicators only for the EuclAB300 instance. For the EuclAB100 instance, PD-TPLS is better on D_2, and for the EuclAB500 instance, PD-TPLS is better on $I_{\epsilon 1}$ and D_2. For the ClusteredAB100 instance, 2PPLS+SpeedP1 finds better results except on D_2. For the ClusteredAB300 instance, PD-TPLS finds better results except on R. For the ClusteredAB500 instance, 2PPLS+SpeedP1 finds better or equivalent results on R and D_1 while PD-TPLS finds better results on $\mathcal{H}$, $I_{\epsilon 1}$ and D_2. The 2PPLS+SpeedP1 method always generates more potentially efficient solutions than PD-TPLS. The running time of PD-TPLS is higher than the running time of 2PPLS+SpeedP1, except on the EuclAB300, ClusteredAB300 and ClusteredAB500 instances where the running time of PD-TPLS is slightly lower. The PD-TPLS method appears thus more competitive than 2PPLS+SpeedP1 on the clustered instances.

Mann-Whitney test

The results of the comparison of 2PPLS+SpeedP1 with PD-TPLS for the Mann-Whitney test are given in Table 8.15, for all the instances, except for the "Kro" instances of low size. The level of risk of the test has been fixed to 1%.

The results show that we can affirm with a very low risk that, for the indicators considered in this work, 2PPLS+SpeedP1 is better or equal to PD-TPLS, except for the ClusteredAB100 instance (on D_2), for the ClusteredAB300 instance (on $\mathcal{H}$, $I_{\epsilon 1}$ and D_1) and for the ClusteredAB500

instance (on $\mathcal{H}$ and $I_{\epsilon 1}$).

Table 8.15: Results of the Mann-Whitney test for the $\mathcal{H}$, $I_{\epsilon 1}$, R, D_1 and D_2 indicators. Comparison between 2PPLS+SpeedP1 and PD-TPLS.

Instance	$\mathcal{H}$	$I_{\epsilon 1}$	R	D_1	D_2
KroAB300	>	>	>	>	>
KroAB400	>	=	>	>	=
KroAB500	>	=	>	>	>
KroAB750	>	>	>	>	=
KroAB1000	>	>	>	>	>
EuclAB100	>	>	>	>	=
EuclAB300	>	>	>	>	>
EuclAB500	>	=	>	>	>
RdAB100	>	>	>	>	>
RdAB300	>	>	>	>	>
RdAB500	>	>	>	>	>
MixedAB100	>	>	>	>	>
MixedAB300	>	>	>	>	>
MixedAB500	>	>	>	>	>
ClusteredAB100	>	>	>	>	<
ClusteredAB300	<	<	=	<	=
ClusteredAB500	<	<	>	=	=

Outperformance relations

We now compare the solutions obtained with 2PPLS+SpeedP1 and PD-TPLS in term of outperformance relations.

We show in Figure 8.21 the results of the comparison between the potentially non-dominated points obtained with 2PPLS+SpeedP1 and with PD-TPLS, for the EuclAB300, RdAB300, MixedAB300 and ClusteredAB300 instances.

The conclusions are the same as in section 8.4.3, since the difference between the quality of the results obtained with 2PPLS and 2PPLS+SpeedP1 is low: many solutions obtained with 2PPLS+SpeedP1 dominate solutions of PD-TPLS, for the EuclAB300, RdAB300 and MixedAB300 instances, except for the ClusteredAB300 instance.

8.7 Conclusion and perspectives

We have shown in this chapter that by using 2PPLS, composed of two different techniques—a very good initial population composed of an approximation of the extreme supported efficient solutions and the Pareto Local Search method—we can obtain very good results for the bTSP. Our method has been compared to another state-of-the-art method for the bTSP, the PD-TPLS method. We have shown that our new method outperforms PD-TPLS for many instances, on both quality of results and running time.

We point up two advantages of 2PPLS:

- Simple method with better results than state-of-the-art results. No numerical parameters is an interesting advantage in compared with multiobjective genetic algorithms which often ask a lot.
- Improving the results is quite easy by using the perturbation algorithm.

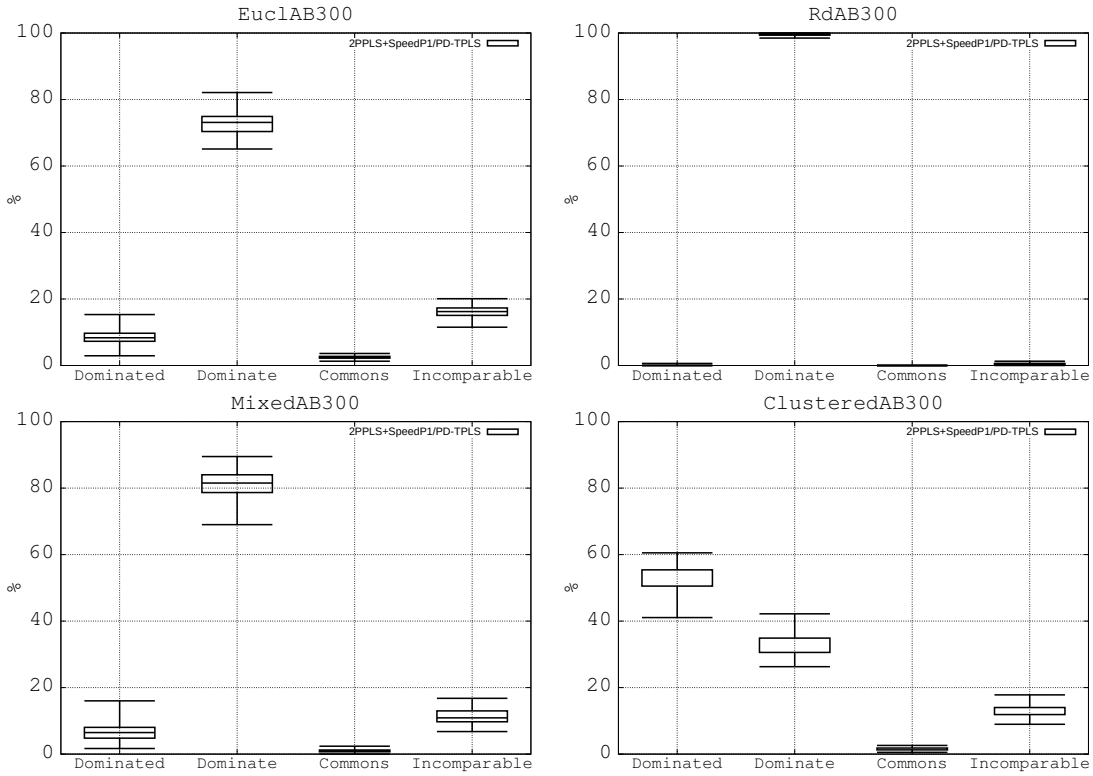

Figure 8.21: Comparison between the potentially non-dominated points obtained with 2PPLS+SpeedP1 and with PD-TPLS for the EuclAB300, RdAB300, MixedAB300 and ClusteredAB300 instances.

A disadvantage of 2PPLS is that the method stops when no more improvement with the considered neighborhood is possible. In some cases, as in the clustered instances, the time before convergence is very high.

We have also proposed different speed-up techniques to solve large-scale bTSP instances with 2PPLS. With the candidate lists we are able to achieve in much shorter time results as good as the standard 2PPLS method with a complete exploration of the neighborhood. With further reduction of the running time, for example if only the edges used by the solutions generated during the first phase are considered in the second phase, the quality of the generated solutions only slightly deteriorates and the quality is statistically comparable for instances with at least 400 cities. Therefore, this simple parameter-free method gives a good compromise between performances and running time. Further reduction of the running time, at the additional cost of the quality, may be obtained by adding the don't look bits technique. Such method may be used to obtain in a short time a reasonably good approximation of the Pareto front of large-scale instances.

We also gave by this work state-of-the-art results for biobjective instances of the TSP with more than 500 cities, until 1000 cities, while before only instances with up to 500 cities have been tackled.

Among the research perspectives, automatically fixing a good value for D for the speed-up technique based on data dominance relations would be relevant, as well as taking into account the frequencies with which the edges are used. Another interesting and obvious future work would be to test 2PPLS on more than two objective instances. A reason why this work has not yet been realized is that the first phase of 2PPLS with three or more objective is more complicated, because the correspondence between the geometrical and the algebraic interpretation of the weight sets is no more valid [194] (see section 3.2.3 in chapter3). Experiments should be done to see if the use of a sophisticated method as first phase will be still worthwhile, or if a simple

method based on randomly or uniformly generated weight sets would be sufficient. The speed-up techniques could also be adapted to other large-scale MOCO problems.

A realization of a better analysis of the behavior of 2PPLS would also be interesting: what solutions are important in the initial population and how to produce initial solutions allowing good improvements with PLS?

Finally, we have only used the two-edge exchange neighborhood in PLS, which is very simple. It is clear that using more complex moves, like the three-edge exchange move or the Lin and Kernighan move would give better results in term of quality, but as the running time would be higher, the speed-up techniques should be carefully adapted to these more complex neighborhoods.

In conclusion, the MOTSP still needs further studies: very few results are known for the TSP with more than two objectives (at the exception of some results for the three-objective case), and no efficient exact method has been developed for solving this problem, even in the biobjective case. Moreover, the instances size for which approximations have been generated with a reasonable running time are still very small compared to the size of the single-objective instances that we can approximate.

9

Multiobjective decomposition of positive integer matrices: application to radiotherapy

We consider in this chapter a multiobjective combinatorial optimization problem that occurs in the treatment of cancer with radiotherapy. We first introduce the radiotherapy treatment and the problem considered (section 9.1). We then give the mathematical formulation of the problem (section 9.2). In the next section, we show how we have solved the problem with 2PPLS (section 9.3). In the section 9.4, we expose the results obtained on random instances and on real instances as well.

9.1 Introduction

In 1896, Wilhelm Conrad Roentgen (who received the first Nobel prize in physics in 1901) reported his discovery of the x-ray (called "x-ray" by Roentgen because it was an unknown type of radiation at this time). It is a form of electromagnetic radiation than can cause cell damage by interfering with the ability of the cells to grow and reproduce [179]. Few months later, along with the parallel discovery of radium by Marie Curie (Nobel prize in physics in 1903), the first cancer patient was irradiated [190]. Indeed, cells that are especially sensitive to radiation are those that reproduce more frequently, such as cancer cells. Normal cells are also affected by radiation but they possess the ability to repair and recover from radiation damage unlike cancer cells. Following this understanding, the medical community adapted radiotherapy to destroy cancer cells.

More than 100 years later, radiotherapy is used by more of the half of people with cancer [235]. Radiotherapy can also be combined with surgery or chemotherapy. Its use depends on the type of the tumor, on the localization of the tumor and on the health of the patient.

The radiotherapy treatment works as follows. A patient lies down on a couch and a linear accelerator, mounted on a gantry, sends radiations in direction of the tumor in order to kill it (see Figure 9.1). The radiations are composed of beams of high energy photons or electrons. The treatment should be such as enough dose is delivered to the targeted region, to kill the cancerous cells, while preserving the surrounding anatomical structures.

Nowadays, intensity-modulated radiation therapy (IMRT) is mainly used (the first clinical IMRT with modern technology was in March 1994 [243]). The IMRT is an advanced type of high-precision radiation, due to the advancements during the 80's and 90's in computer technology, electronic miniaturization, three-dimensional digital imaging and specialized software tools [190]. IMRT allows to optimally conforms radiation beams to tumor shapes thanks to three-dimensional imaging and a device called multileaf collimator (MLC).

The MLC is composed of a set of leaves, made of usually tungsten, that can move independently in and out of the path of a beam in order to block it (see Figure 9.2). The role of the MLC is to modulate the radiations. Indeed, by combining different configurations of the MLC, it is possible to obtain different intensities to reduce radiation doses delivered to the surrounding

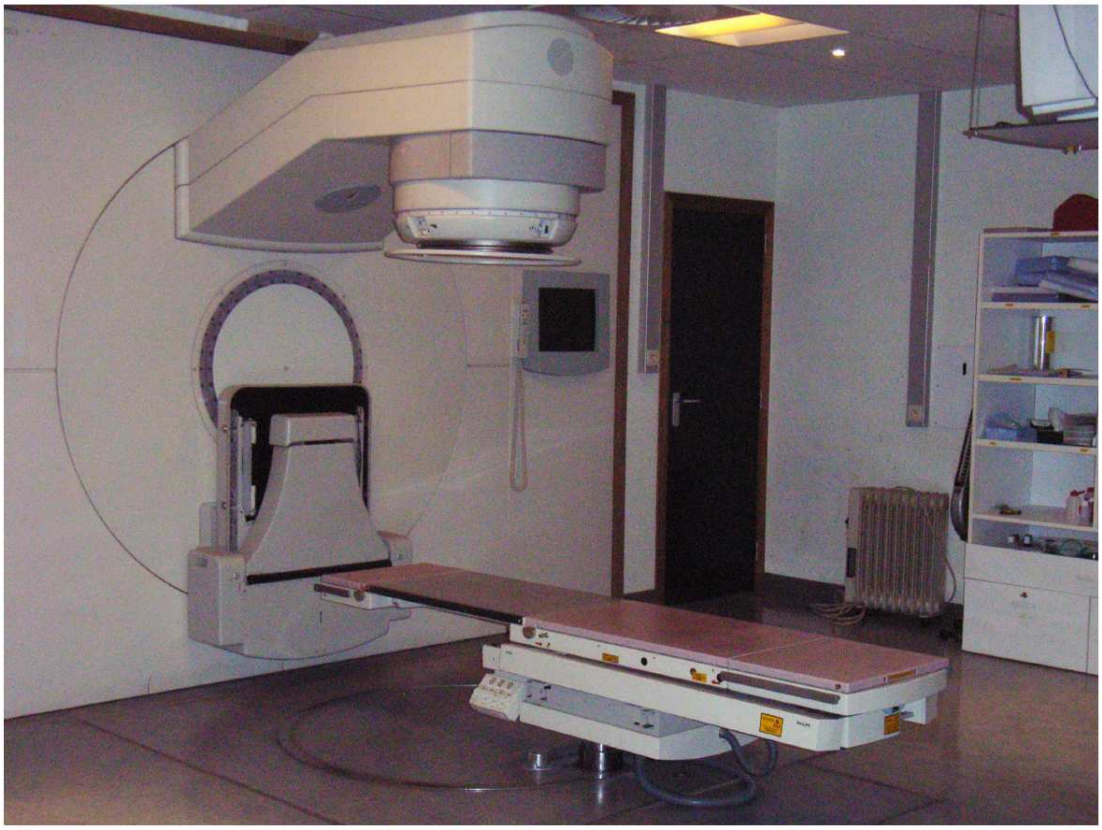

Figure 9.1: Radiotherapy room of the university hospital of Saint Luc (Brussels) [66].

healthy organs and tissues. In Figure 9.3, we can see the comparison between conventional beam profile and IMRT beam profile.

Also, by moving the gantry or the couch to treat the tumor from various angles, it is possible to prescribe a dose across all three dimensions (see Figure 9.4).

Figure 9.2: A multileaf collimator, allowing to build different shapes of radiations (source: Varian medical system, `http://www.varian.com`).

Some advantages of IMRT are the following [235]:

- Fewer side effects and higher and more effective radiation doses.
- Concave organs treatment possibility.

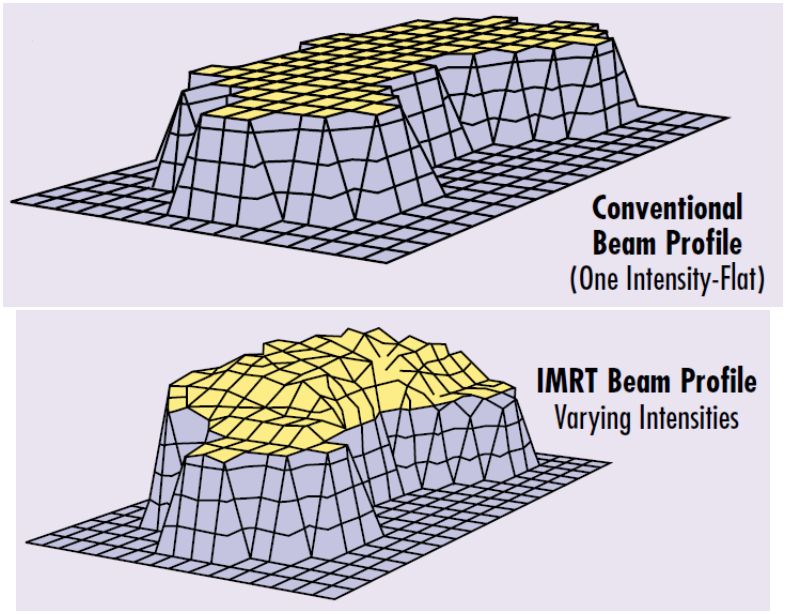

Figure 9.3: Comparison between conventional beam profile and IMRT beam profile [190].

Figure 9.4: The gantry and the couch can move in order to treat the tumor from various angles (source: the radiation therapy and cancer institute, `http://www.rtcionline.com`).

- Better dose repartition.
- Cure rates in cancer patients can be improved by 15-40 percent in comparison with conventional radiotherapy.

On the other hand, IMRT makes the treatment more complex and demands new skills for the physician, dosimetrist and physicist. It can take as long as 30 minutes to give an IMRT treatment, versus 5-10 minutes for conventional radiotherapy. Also, a radiation oncology department may need to invest millions (about 7 millions € for a new operational unit [235]) to get a new IMRT machine, planning software, and increased personnel. Moreover, it takes more manpower to design a radiation plan and to deliver the treatment.

In Belgium, the situation was the following in 2007. Amongst the 24 registered radiotherapy centers in Belgium, 12 centers use the IMRT technology. Eleven other centers planned to develop this activity soon and only one did not plan to install IMRT in the coming years [235].

The evolution of the number and percentage of patients treated in Belgium are shown in Tables 9.1 and 9.2 [235].

Table 9.1: Evolution of the number of patients treated by IMRT in Belgium.

Year	Number of patients
2004	760
2005	1411
2006	2154
2007	2795

Table 9.2: Percentage of IMRT patients of all patients treated with radiotherapy (in the centers that offer IMRT).

Year	Percentages of IMRT patients
2001	2.1
2002	2.6
2003	2.7
2004	5.6
2005	9.3
2006	15.0
2007	20.8

Treatment planning is achieved in most systems using inverse planning software algorithms. First, imaging techniques (computed tomography, magnetic resonance imaging, positron emission tomography,...) are used to diagnose and localize tumors. Images of both target tissues and tissues at risk are required to define for the doses upper bounds to healthy organs and lower bounds to the target. Then the IMRT optimization process determines the beam parameters that will lead to the desired solution.

The IMRT optimization process is usually composed of three phases [59]:

- The selection of the number of beams and the beam angles through which radiation is delivered (geometry problem);
- The computation of an optimal intensity map for each selected beam angle (intensity problem);
- The determination of a sequence of configurations of the MLC (realization problem).

Even if these problems are connected, they are often solved independently (that can however only gives suboptimal overall results).

The geometry and intensity problems are very closed. In these problems, the aim is to find locations of beams and beam intensities in order to achieve a high uniform dose in the target volume to increase tumor control, while limiting the dose to healthy organs and tissues as much as possible to reduce the probability of complications (see Figure 9.5).

There exist multiobjective formulations of these problems, since both criteria, that is high radiation in target volume and low radiation in organs at risk are conflicting. For a review of the literature of these two problems, we refer the reader to the recent and interesting survey of Ehrgott *et al.* [59].

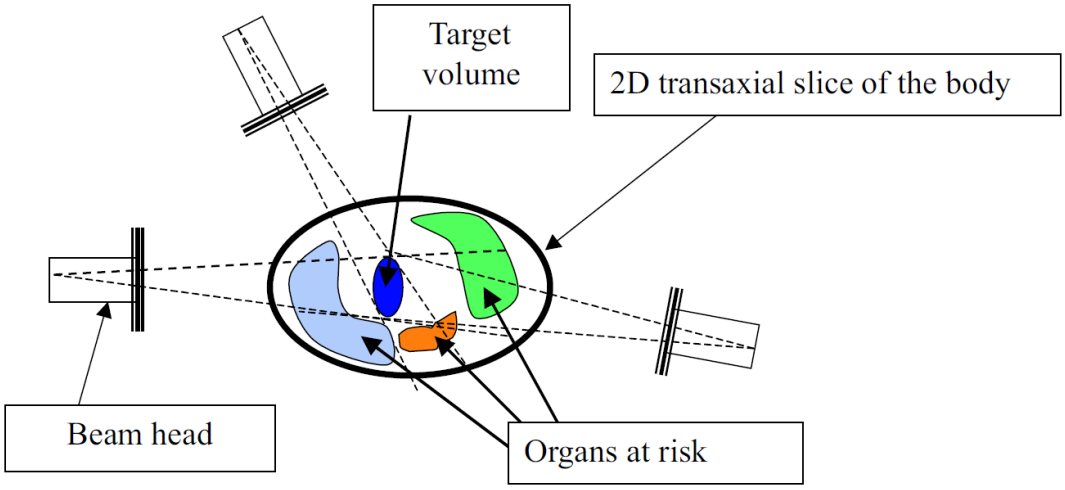

Figure 9.5: The geometry and intensity problems: how many beam angles, where and how much radiation to send in order to get high radiation in target volume and low radiation in organs at risk [11].

We only consider in this work the realization problem. We thus make the assumption that the beam angles are fixed and that for each beam angle, the intensity matrix is given (which is not constant since inhomogeneous dose levels are administrated: certain cancer targets receive a required amount of dose while functional organs are spared). We are in the situation of Figure 9.6. In this figure, the intensity matrices are represented with grayscale coded grids (the whiter the square, the higher the value of the radiation in the square is).

We present in the next section the mathematical formulation of the realization problem, that is how to realize the intensity matrices with the MLC.

9.2 The realization problem

Throughout we use the notation $[n] := \{1, 2, \cdots, n\}$ for positive integers n.

We consider a positive integer matrix A of size $m \times n$: $A = (a_{i,j})$ with $i \in [m]$ and $j \in [n]$. The matrix corresponds to the intensity matrix giving the different values of radiation that the linear accelerator has to send throughout the body of a patient. The value $a_{i,j}$ of A gives the desired intensity that should be delivered to the coordinate (i, j).

We have to decompose the matrix A into a set of segments. The segments correspond to the shape of the MLC. In the MLC, we make the distinction between two types of leaves: the left leaves that move from the left to the right and the right leaves that move from the right to the left.

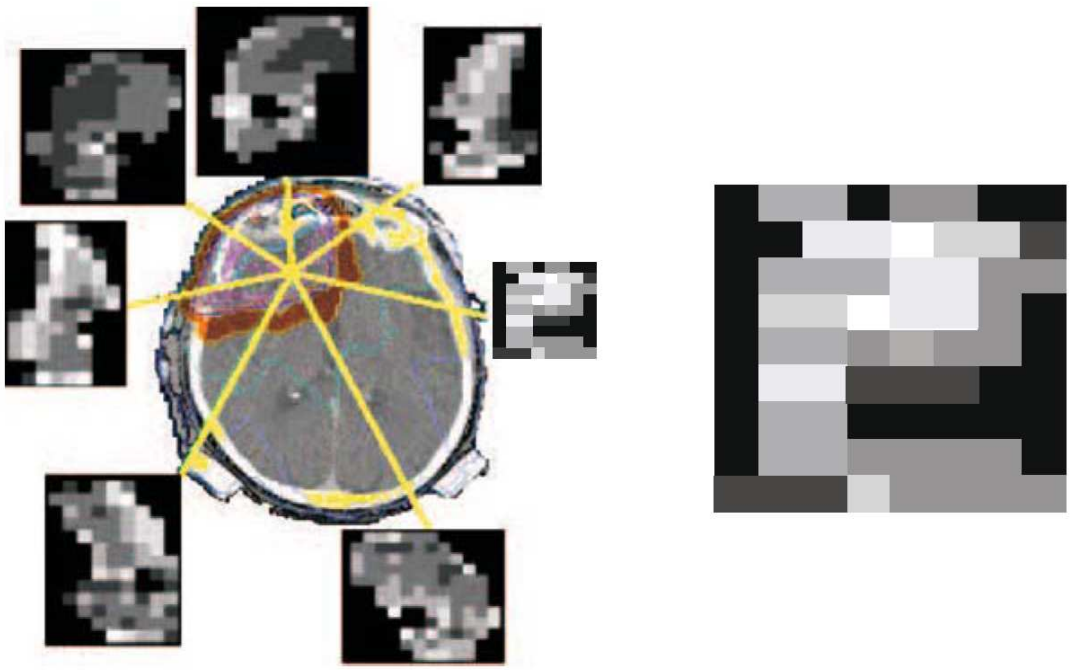

Figure 9.6: IMRT with seven beam angles. The intensity matrices are represented with grayscale coded grids [206].

A segment can be represented by a special binary matrix of size $m \times n$ that describes the left and right leaves positions. These matrices have to respect the consecutive ones property (C1), which means, in short, that the ones occur consecutively in a single block in each row (since we can only block the radiations with a left or a right leaf).

A segment is noted $S = (s_{i,j})$ with $i \in [m]$ and $j \in [n]$. An example of a decomposition of a matrix A into segments is shown below.

$$\begin{aligned} A &= \begin{pmatrix} 4 & 8 & 3 \\ 5 & 2 & 1 \\ 5 & 7 & 2 \end{pmatrix} \\ &= 4\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} + 2\begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} + 1\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} + 1\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \end{aligned}$$

The positions of the left and right leaves corresponding to a segment S are given by the l_i and r_i integers defined as follows:

$$0 \leq l_i < r_i \leq n+1 \quad (i \in [m])$$

$$s_{i,j} = \begin{cases} 1 & \text{if } l_i < j < r_i \quad (i \in [m], j \in [n]) \\ 0 & \text{otherwise.} \end{cases}$$

We denote the set of feasible segments by $\mathcal{S}$.

For example, for the following segment:

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

we have: $l_1 = 0, r_1 = 3; l_2 = 0, r_2 = 2$ and $l_3 = 2, r_3 = 4$.

It should be noted that in this work, a line full of zero will always be defined with $l = 0$ and $r = 1$.

A feasible decomposition of A is a linear sum of segments and has the following form:

$$A = \sum_{k=1}^{K} u_k S^k \text{ with } u_k \in \mathbb{N}_0, S^k \in \mathcal{S},\ \forall\, k \in [K].$$

Two criteria are generally considered to evaluate the quality of a decomposition: the total irradiation time and the setup-time.

The total irradiation time, very important from a medical point of view, is the time during which a patient is irradiated. This criterion is proportional to the sum of the coefficients (**decomposition time**).

The setup-time is the time needed to configure the MLC. This criterion is generally considered as proportional to the number of segments (**decomposition cardinality**), with constant times to move from one segment to the next. It is important to minimize this criterion in order to reduce the time of the session and so the comfort of the patient. Moreover, an optimal scheduling for radiotherapy is more and more required since the number of patients treated by radiotherapy increases. It is only recently that this scheduling problem has caught the attention of the operations research community [34, 131]. Obviously, minimizing the time of the session will improve the quality of the scheduling and the number of patients treated.

We can formulate both objectives that is the decomposition time (DT) and the decomposition cardinality (DC) as follows:

$$\textbf{(DT)}\quad \min\left\{ \sum_{k=1}^{K} u_k \;\middle|\; A = \sum_{k=1}^{K} u_k S^k, u_k \in \mathbb{N}_0, S^k \in \mathcal{S},\ \forall\, k \in [K] \right\}$$

$$\textbf{(DC)}\quad \min\left\{ K \;\middle|\; A = \sum_{k=1}^{K} u_k S^k, u_k \in \mathbb{N}_0, S^k \in \mathcal{S},\ \forall\, k \in [K] \right\}$$

Polynomial algorithms are known for the DT minimization [24, 210]. Many optimal solutions can be found for this single-objective problem.

On the other hand, the DC minimization has been proved to be $\mathcal{NP}$-Hard [12, 28].

Some authors [12, 65, 130] optimize the objectives lexicographically: by first minimizing DT and then trying to reduce DC while keeping the minimal value for DT. Taşkin *et al.* [216] and Wake *et al.* [241] recently considered both objectives at the same time, but by simply doing a linear sum of the objectives.

To be more realistic, we do not only consider in this work constant times to move from one segment to the next. The **variable setup-time** is defined as follows:

$$(\mathbf{SU}_{\boldsymbol{var}})\quad \min\left\{ \sum_{k=1}^{K-1} \mu(S^k, S^{k+1}) \;\middle|\; A = \sum_{k=1}^{K} u_k S^k, u_k \in \mathbb{N}_0, S^k \in \mathcal{S},\ \forall\, k \in [K] \right\}$$

where μ is proportional to the time necessary to change the setup of the MLC from the configuration corresponding to S^k to the configuration corresponding to S^{k+1}. This objective is also known under the name overall leaf travel time [130].

The value μ between two segments S^k and S^{k+1} is computed as follows [60]:

$$\mu\big(S^k, S^{k+1}\big) = \max_{1 \le i \le m} \max\left\{ \left|l_i^{k+1} - l_i^k\right|, \left|r_i^{k+1} - r_i^k\right| \right\}$$

It thus corresponds to the maximal move that a leaf has to make to attain its next position. Between two different segments, the minimal value that can take μ is equal to one.

The overall leaf travel time objective does not have to be neglected since the velocity of the leaves is limited due to precision issues. Moreover, the more the leaves have to move, higher the risk of imprecision is [256].

Once the segments are fixed, the minimization of the overall leaf travel time is equivalent to a search for a Hamiltonian *path* of minimal cost on the complete graph which has the segments as vertices and the cost function μ on the edges (see Figure 9.7). The cost function μ has the property to be a metric [130].

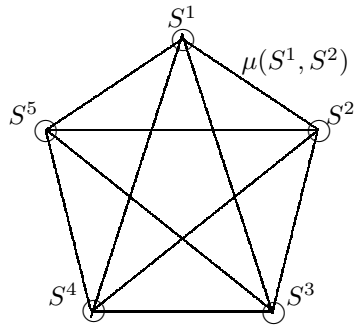

Figure 9.7: Minimization of the overall leaf travel time by searching a Hamiltonian path of minimal cost on the complete graph which has the segments as vertices and the cost function μ on the edges.

This problem can be transformed to a TSP problem (Hamiltonian *cycle* of minimal cost) by adding a dummy vertex which has a distance of zero to all other vertices. However, with this transformation, the triangular inequality in Euclidean space no longer holds which makes the TSP problem a little bit harder to solve.

The key of the minimization of the overall leaf travel time is how to generate segments that will give a solution with a Hamiltonian path of minimal value, since once the segments are fixed, the minimization of the overall leaf travel time is relatively easy.

As there is a positive correlation between DC and SU_{var}, the few authors that have considered SU_{var} tried to first minimize DC and then SU_{var} by generating the best sequence of segments. Kalinowski [130] uses a minimum spanning tree approximation to find the best sequence of segments, but it also possible to use an exact TSP algorithm, as done by Ehrgott *et al.* [60]. Siochi [210] considered both objectives at the same time, but through a linear sum.

In this work, we will consider the three objectives (DT,DC,SU_{var}) simultaneously. The multiobjective formulation of the problem (P) considered is thus as follows:

$$
(\mathrm{P}) \begin{cases} \min f_1(x) & = & \displaystyle\sum_{k=1}^{K} u_k \quad (\mathbf{DT}) \\ \min f_2(x) & = & K \quad (\mathbf{DC}) \\ \min f_3(x) & = & \displaystyle\sum_{k=1}^{K-1} \max_{1 \le i \le m} \max \left\{ \left| l_i^{k+1} - l_i^k \right|, \left| r_i^{k+1} - r_i^k \right| \right\} \quad (\mathbf{SU_{var}}) \\ \text{subject to } A & = & \displaystyle\sum_{k=1}^{K} u_k S^k, u_k \in \mathbb{N}_0, S^k \in \mathcal{S}, \ \forall\, k \in [K] \end{cases}
$$

We denote by $\mathcal{X}$ the feasible set in decision space, defined by $\mathcal{X} = \left\{ x \in \{(u_k \in \mathbb{N}_0, S^k \in \mathcal{S})\}^K | A = \sum_{k=1}^{K} u_k S^k \right\}$. The feasible set in objective space is called $\mathcal{Y}$ and is defined by $\mathcal{Y} = f(\mathcal{X}) = \left\{ (f_1(x), f_2(x), f_3(x)), \forall\, x \in \mathcal{X} \right\} \subset \mathbb{Z}^3$.

To our knowledge, this multiobjective problem has never been tackled before: nobody has tried to find the efficient solutions, or even a good approximation of the efficient solutions of P. Our aim is to generate a good approximation of a minimal complete set of the efficient solutions of P.

We have created the following example in order to show that the objectives can be conflicting.

The matrix A to decompose is the following:

$$A = \begin{pmatrix} 3 & 2 & 3 \\ 2 & 5 & 1 \end{pmatrix}$$

Let us consider the three following decompositions:

- D_1:
$$1\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} + 1\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} + 1\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} + 2\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$
with DT=5, DC=4 and SU_{var}=(1+1+2)=4. It is impossible to improve the SU_{var} value of this solution since the distance between the last segment and the other segments is always equal to 2.

- D_2:
$$2\begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} + 3\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + 1\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$
with DT=6, DC=3 and SU_{var}=(2+2)=4. It is impossible to improve the SU_{var} value of this solution since the distance between each segment is always equal to 2.

- D_3:
$$3\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + 2\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} + 2\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} + 1\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$
with DT=8, DC=4 and SU_{var}=(1+1+1)=3.

Property 1 *The optimal value of DT for the matrix A is equal to 5.*

Proof 1 See Engel's property (Property 7) in the next section.

The solution D_1 is thus an optimal solution for DT.

Property 2 *The optimal value of DC for the matrix A is equal to 3.*

Proof 2 Suppose that a decomposition with two segments exists, with two different coefficients a and b. By adding these two segments, we can generate a matrix with at most four different numbers equal to $(0, a, b, a + b)$. As the matrix A contains four different numbers equal to $(1, 2, 3, 5)$ but not zero, it is impossible to find a decomposition with two segments.

The solution D_2 is thus an optimal solution for DC.

Property 3 *The solution D_2 is the unique optimal solution for DC.*

Proof 3 Because of the C1 constraint, the production of the two 3 of the matrix A has to be realized with two different combinations of segments. Given that only three segments can be used, there are two possibilities: the first one is to use two segments with coefficients respectively equal to 2 and 1 and one segment with a coefficient 3, as done in the solution D_2. The other possibility is to use one segment with a coefficient 3 and another with a coefficient 3. In this case, the only possibility to produce the 5 of the matrix A, is to use a segment with a coefficient equal to 2 or 5. But that would make impossible the generation of the 1. Therefore, the decomposition D_2 is the only solution with DC equal to 3.

Property 4 *For the matrix A, it is impossible to find a solution optimal for DT and DC at the same time.*

Proof 4 Directly from Property 3, since the solution D_2 is the unique optimal solution for DC and this solution is not optimal for DT. Another proof is the following. Let us enumerate the different possibilities of coefficients for a solution that would have an optimal value of DT equal to 5 and an optimal value of DC equal to 3. The first possibility would be to use the coefficients (3,1,1). In this case, in order to produce the two 3 of the matrix A, the only possibility is to use the coefficient 3 since we cannot produce a 3 with two 1. Although, it is impossible to produce the two 3 at the same time because of the C1 constraint. The second possibility would be to use the coefficients (2,2,1). But in this case, after using one or both segments with a coefficient 2, the first line of the remaining matrix would be equal to (1 0 1). This remaining matrix cannot be decomposed with only one segment with a coefficient equal to 1 (because of the C1 constraint). As there are no other possibilities, we can conclude that is impossible to find a decomposition optimal for DC and DT at the same time for the matrix A.

Property 5 *The optimal value of SU_{var} for the matrix A is equal to 3.*

Proof 5 Suppose that a decomposition with SU_{var} equal to 2 exists. This decomposition has to be composed of at most three segments. But as it is impossible to obtain a decomposition with less than three segments (see Property 2), this decomposition has to be composed of three segments. As the decomposition D_2 is the only solution with DC equal to 3 and that the SU_{var} value of D_2 is equal to 4 and cannot be improved by changing the sequence, a decomposition with SU_{var} equal to 2 does not exist. SU_{var} equal to 3 is thus the optimal value.

The solution D_3 is thus an optimal solution for SU_{var}.

We can wonder whether a solution optimal for DT and SU_{var} at the same time exists, that is with a DT value equal to 5 and a SU_{var} value equal to 3.

Property 6 *For the matrix A, it is impossible to find a solution optimal for DT and SU_{var} at the same time.*

Proof 6 As consequence of Properties 3 and 4, an optimal solution for DT cannot be composed of three segments. Therefore, a decomposition optimal for DT and SU_{var} should be composed of four segments (if the number of segments is higher than four, the SU_{var} value would be necessarily superior to 3). But the only four coefficients a, b, c and d ($\in \mathbb{N}_0$) that respect $(a+b+c+d=5)$ are $(2,1,1,1)$, that is the coefficients of the decomposition D_1. As D_1 holds a SU_{var} value equal to 4 that cannot be improved, it is impossible to find a solution which is optimal for DT and SU_{var} at the same time for the matrix A.

We have thus shown, for this example, that one solution optimizing DT has not an optimal value for DC or SU_{var} and that the unique solution optimizing DC has not an optimal value for DT or SU_{var}. And therefore, a solution that optimizes SU_{var} has not an optimal value for DT or DC. The objectives of this example are thus conflicting. As far as we know, this is the smallest example in size (3x2) and also for the maximal value of the matrix (5) that holds this property.

The three non-dominated points of this example are (5,4,4), (6,3,4) and (8,4,3) (it is easy to check that we cannot find a non-dominated point with DT=7).

9.3 Adaptation of 2PPLS to the multiobjective decomposition problem

In this section, we present how 2PPLS (see chapter 6, section 6.3) has been adapted to the problem P. This method has been selected for its simplicity and efficiency. We first present how the initial population has been generated. We expose then the neighborhood needed in PLS, method used as second phase in 2PPLS.

9.3.1 Initial population

Two initial solutions of good quality are generated and used to form the initial population. The first solution is a good approximation of lexmin(DT,DC,SU_{var}) and the second one is a good approximation of lexmin(DT,SU_{var},DC). In both cases, we first minimize DT since polynomial algorithms are known for this problem. We can remark that a solution corresponding to lexmin(DT,SU_{var}) is also a solution corresponding to lexmin(DT,SU_{var},DC), since once SU_{var} is minimized, the DC value is set and cannot be modified.

Approximation of lexmin(DT,DC,SU_{var})

To approximate lexmin(DT,DC,SU_{var}), we first approximate lexmin(DT,DC) with the heuristic of Engel [65] which is efficient for this problem. We then apply a TSP heuristic to reduce the SU_{var} value of the solution. We use a very efficient heuristic for the TSP: the Lin and Kernighan heuristic implemented by Helsgaun (LKH) [103].

The algorithm developed by Engel is a deterministic construction algorithm, that allows to find an optimal solution for the DT objective with a low DC value. Engel tackles this problem as follows: he removes different well-selected combinations of couples (u_t, S^t) from the current matrix until $A_{t+1} = 0$, with $A_{t+1} = A_t - u_t S^t$, where t represents the index of the step of the construction algorithm. He starts the algorithm with $A_0 = A$. A move consists thus of removing from the current matrix a segment multiplied by a certain coefficient.

At each step of the construction heuristic, the maximum coefficient (u_{max}) that can be used while ensuring that the optimal objective DT can be achieved is considered. Using u_{max} allows to find very good results for the lexicographic problem (DT,DC) [60, 65, 130], since using decompositions with high coefficient values increases the chances to find a solution with a low value of DC. Unfortunately, no good quality upper bound for the DC values obtained with this heuristic has been found.

Engel has developed a theory to compute u_{max}. The coefficient u_{max} can be easily obtained in $O(mn^2)$.

We present below the basic results of Engel on which are based the computation of the coefficient u_{max} (we do not go into details).

If two zero columns are added to A, that is let:

$$a_{i,0} = a_{i,n+1} = 0 \quad \forall\, i \in [m]$$

we can associate to A its difference matrix D of dimension $m \times (n+1)$:

$$d_{i,j} = a_{i,j} - a_{i,j-1} \quad \forall\, i \in [m],\ \forall j \in [n+1]$$

The row complexity $c_i(A)$ of A is defined by

$$c_i(A) = \sum_{j=1}^{n} \max\{0, d_{i,j}\}$$

and the complexity $c(A)$ of A is the maximum of the row complexity:

$$c(A) = \max_{i \in [m]} c_i(A)$$

Property 7 ***(Engel's property)*** *The complexity $c(A)$ is equal to the optimal DT value that we can obtain for a decomposition of A* [65].

For instance, for the matrix of the preceding section:

$$\begin{pmatrix} 3 & 2 & 3 \\ 2 & 5 & 1 \end{pmatrix}$$

the difference matrix is equal to:

$$D = \begin{pmatrix} 3 & -1 & 1 & -3 \\ 2 & 3 & -4 & -1 \end{pmatrix}$$

The complexity c_1 of the first row is equal to 4 while the complexity c_2 of the second row is equal to 5. The complexity of the matrix is thus equal to 5 ($= \max(4, 5)$), which is equal to the optimal value for DT.

In the heuristic of Engel, to keep the optimal DT value, the coefficient u_{max} has to respect the following properties:

$$c(A - u_{max}S) = c(A) - u_{max}, \quad \text{that is}$$

$$c_i(A - u_{max}S) \leq c(A) - u_{max} \quad \forall\, i \in [m]$$

Engel gives a $O(mn^2)$ algorithm to compute u_{max} based on the preceding relations (the algorithm is not detailed here).

But once u_{max} has been defined, we also have to define which segments to use among all segments that respect u_{max}. Kalinoswki [130] has developed a trial and error rule that gives slightly better results that the initial rule of Engel. The rule is as follows.

He computes the scalar q associated to A as following:

$$q(A) = \left|\{(i, j) \in [m] \times [n] : d_{i,j} \neq 0\}\right|,$$

and in his method, a segment S that minimizes $q(A - uS)$ is selected.

This method gives very good results for lexmin(DT,DC), but these results are only based on experiments and not on a theoretical proof.

Approximation of lexmin(DT,SU$_{var}$)

To approximate lexmin(DT,SU$_{var}$), we propose to adapt the heuristic of Engel since SU$_{var}$ is correlated to the DC objective. We keep the principle of the construction algorithm of Engel by removing well-selected combinations of couples (u_t, S^t) from the current matrix until $A_{t+1} = 0$. As in the Engel algorithm, it is worthwhile to take the maximal coefficient u_{max} that allows to keep the minimal DT value, since the SU$_{var}$ objective is linked to the DC objective. For the definition of the segment that corresponds to u_{max}, we try three new rules. These three new rules are presented below.

For each line of the new segment to define, we have to choose between different (l_i, r_i) intervals. We sort the intervals as follows: $\{(0,1), (0,2), \cdots, (0,n+1), (1,3), \cdots, (1,n+1), \cdots, (n-2,n), (n-2,n+1), (n-1,n+1)\}$ (see Figure 9.8 for an example for a line of length 4) and we define three rules from this sorting:

- The first rule is to take the first feasible interval (that is an interval that does not irradiate too much an entry of the intensity matrix to decompose), following this order.
- The second rule is to take the last feasible interval, following this order.
- The third rule is to take the first feasible interval which is the closest to the preceding interval of the same line (that is minimizing $\max\left\{\left|l_i^{k+1} - l_i^k\right|, \left|r_i^{k+1} - r_i^k\right|\right\}$): the aim is to minimize the maximal distance between two consecutive segments. The first interval defined with this rule is the same as the one selected with the first rule.

For the first two rules, we expect to always move the leaves from the left to the right or to the right from the left in order to minimize the maximal distance between two consecutive segments.

The results of the comparison between the different rules will be given in section 9.4.

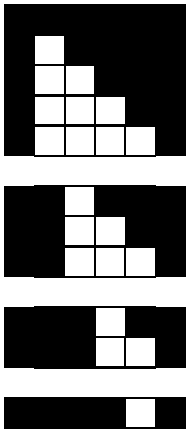

Figure 9.8: Specific order of the leaves for a line of length 4.

9.3.2 Neighborhood

The neighborhood is the main component of PLS (employed in the second phase of 2PPLS). On the other hand, it is not trivial to produce neighbors from a feasible solution for the problem P considered. Removing one segment from the current solution is not enough to produce a solution different than the current solution. Removing two segments from the current solution requires to determine how to select both segments and how to recombine these two segments to produce a new feasible solution, ideally of better quality. Moreover, it will be difficult with this kind of technique to find neighbors with different DC values. We can try to modify the sequence of segments in order to improve the SU_{var} objective, but it is always possible to apply a TSP algorithm at the end of the decomposition to improve this objective.

The neighborhood developed in this work is thus a bit complex. It works as follows, for the generation of one neighbor from a current decomposition:

1. Selection of a segment S that belongs to the current decomposition.
2. We modify a line i of S in the following way:

$$\begin{aligned} l_i &= l_i + (-1 \text{ or } 0 \text{ or } 1) \\ r_i &= r_i + (-1 \text{ or } 0 \text{ or } 1) \end{aligned}$$

3. We put S at the first place of the new decomposition.
4. We eventually modify the coefficient of this segment.
5. We construct a neighbor by adding the segments in the order of the current decomposition. If a segment is not feasible, we skip it. We adapt the coefficient of the segments added, which is equal to the minimum between the current coefficient of the segment and the maximal feasible coefficient. This rule has been adopted since the current coefficient comes from a preceding solution of good quality.
6. The matrix that remains after adding all possible segments is decomposed with the heuristic of Engel.
7. Once a decomposition is obtained, we optimize the SU_{var} objective by using a simple and fast TSP heuristic: the first improvement local search based on the two-edge exchange moves [102].

This neighborhood requires the definition of many elements, but we will see that it is possible to explore many possibilities, in order to produce many neighbors.

To illustrate the neighborhood, we show its functioning on the following example:

$$B = \begin{pmatrix} 8 & 5 & 6 \\ 5 & 3 & 6 \end{pmatrix}$$

The difference matrix of B is equal to:

$$D = \begin{pmatrix} 8 & -3 & 1 & -6 \\ 5 & -2 & 3 & -6 \end{pmatrix}$$

The complexity c_1 of the first row is equal to 9 while the complexity c_2 of the second row is equal to 8. The complexity of the matrix is thus equal to 9 (=max(8, 9)), which is equal to the optimal value for DT.

Let us consider that we start the neighborhood from one of the solutions that minimize the DT objective:

$$3\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 1\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} + 3\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} + 2\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

The DT value of this solution is optimal and equal to (3+1+3+2)=9. DC is equal to 4 and SU_{var} is equal to (2+2+2)=6.

We apply the neighborhood to this solution:

1. We select the first segment:
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
2. We select the second line. For this line, $l = 2$ and $r = 4$. We modify this line by putting $l = 1$, $r = 3$. We obtain this segment:
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$
3. We put this segment at the first place of the new decomposition.
4. The current coefficient of this segment is equal to 3. As for this segment, the maximal feasible coefficient that we can put is 3, we keep the coefficient equal to 3. The remaining matrix is:
$$\begin{pmatrix} 5 & 5 & 6 \\ 5 & 0 & 6 \end{pmatrix}$$
5. The first segment that we can consider is:
$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$
but we cannot add it.
The second segment is:
$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$
We add it, with a coefficient equal to 3, that is its current coefficient (the maximal feasible coefficient is equal to 5). The remaining matrix is:
$$\begin{pmatrix} 2 & 2 & 3 \\ 2 & 0 & 6 \end{pmatrix}$$
The last segment in the initial decomposition is:
$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$
but we cannot add it.

6. The decomposition of the remaining matrix with the heuristic of Engel gives:

$$5\begin{pmatrix}0&0&0\\0&0&1\end{pmatrix}+2\begin{pmatrix}1&1&1\\1&0&0\end{pmatrix}+1\begin{pmatrix}0&0&1\\0&0&1\end{pmatrix}$$

The decomposition obtained is thus:

$$3\begin{pmatrix}1&0&0\\0&1&0\end{pmatrix}+3\begin{pmatrix}1&1&1\\1&0&0\end{pmatrix}+5\begin{pmatrix}0&0&0\\0&0&1\end{pmatrix}+2\begin{pmatrix}1&1&1\\1&0&0\end{pmatrix}+1\begin{pmatrix}0&0&1\\0&0&1\end{pmatrix}$$

But as we can see, it is possible to combine the second segment with the fourth one, to obtain this decomposition:

$$3\begin{pmatrix}1&0&0\\0&1&0\end{pmatrix}+5\begin{pmatrix}1&1&1\\1&0&0\end{pmatrix}+5\begin{pmatrix}0&0&0\\0&0&1\end{pmatrix}+1\begin{pmatrix}0&0&1\\0&0&1\end{pmatrix}$$

Therefore, each time we try to add a segment to a current decomposition, we check if we can combine this segment with other segments of the decomposition, in order to reduce the DC and SU_{var} values of the decomposition.

The DT value of this solution is equal to (3+5+5+1)=14, DC is equal to 4 and SU_{var} is equal to (2+3+3)=8.

7. The matrix of distances between the segments is as follows:

$$\begin{bmatrix}&S_1&S_2&S_3&S_4\\S_1&0&2&1&2\\S_2&2&0&3&2\\S_3&1&3&0&3\\S_4&2&2&3&0\end{bmatrix}$$

We see that by doing a two-edge exchange move, we can obtain the new sequence (S_3,S_1,S_2,S_4), that improves the SU_{var} objective by 3 units. We obtain the following neighbor:

$$5\begin{pmatrix}0&0&0\\0&0&1\end{pmatrix}+3\begin{pmatrix}1&0&0\\0&1&0\end{pmatrix}+5\begin{pmatrix}1&1&1\\1&0&0\end{pmatrix}+1\begin{pmatrix}0&0&1\\0&0&1\end{pmatrix}$$

The evaluation vector of this solution is thus equal to (14,4,5).

We have therefore obtained a new potentially efficient solution, and we can again apply the neighborhood from this solution:

1. We select the third segment:
$$\begin{pmatrix}1&1&1\\1&0&0\end{pmatrix}$$
2. We select the first line. For this line, $l = 0$ and $r = 4$. We modify this line by putting $r = 3$. We obtain this segment:
$$\begin{pmatrix}1&1&0\\1&0&0\end{pmatrix}$$
3. We put this segment at the first place of the new decomposition.
4. The current coefficient of this segment is equal to 5. As for this segment, the maximal feasible coefficient that we can put is 5, we keep the coefficient equal to 5. The remaining matrix is:
$$\begin{pmatrix}3&0&6\\0&3&6\end{pmatrix}$$

5. The first segment that we can consider is:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

We add it, with a coefficient equal to 5, that is its current coefficient (the maximal feasible coefficient is equal to 6). The remaining matrix is:

$$\begin{pmatrix} 3 & 0 & 6 \\ 0 & 3 & 1 \end{pmatrix}$$

The second segment is:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

We add it, with a coefficient equal to 3, that is its current coefficient (the maximal feasible coefficient is equal to 3). The remaining matrix is:

$$\begin{pmatrix} 0 & 0 & 6 \\ 0 & 0 & 1 \end{pmatrix}$$

The last segment in the initial decomposition is:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

We add it, with a coefficient equal to 1, that is its current coefficient (the maximal feasible coefficient is equal to 1).

6. The remaining matrix is:

$$\begin{pmatrix} 0 & 0 & 5 \\ 0 & 0 & 0 \end{pmatrix}$$

The decomposition of this matrix with the heuristic of Engel gives:

$$5\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

The decomposition obtained is thus:

$$5\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} + 5\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 3\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + 1\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} + 5\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

But as we can see, it is possible to combine the second segment with the last one, and then the result of this combination with the fourth segment to obtain this decomposition:

$$5\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} + 3\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + 6\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

with a SU_{var} value equal to (1+2)=3.

7. The matrix of distances between the segments is as follows:

$$\begin{bmatrix} & S_1 & S_2 & S_3 \\ S_1 & 0 & 1 & 2 \\ S_2 & 1 & 0 & 2 \\ S_3 & 2 & 2 & 0 \end{bmatrix}$$

We see that it is impossible to improve the SU_{var} objective by doing two-edge exchange moves.

The DT value of this solution is equal to (5+3+6)=14, and the DC and SU_{var} values are equal to 3 (which are the optimal values of these two objectives [60]).

Therefore, by applying two times the neighborhood from a solution that minimizes the DT objective, we have found a solution that minimizes the DC and SU_{var} values.

It should be noted that we have found one more non-dominated point for this problem: the point (10,4,4), that can be obtained with this solution:

$$2\begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} + 3\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + 1\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} + 4\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

For this small problem, we have therefore found three non-dominated points: (9,4,6), (10,4,4) and (14,3,3).

9.3.3 Final optimization step

For each potentially efficient solution found at the end of 2PPLS, we apply the LKH heuristic, to eventually improve the SU_{var} value of the solutions.

9.4 Results

9.4.1 Instances

We use two types of instances: random instances and real instances. The random instances are the same instances than Engel and Kalinowski used to test their algorithm for lexmin(DT,-DC). That allows us to check that our implementation of the heuristic of Engel gives the same results. The random instances are matrices 15x15 with randomly generated elements (uniformly distributed) between zero and the parameter L (L varies from 3 to 16 as done by Engel and Kalinowski).

The real instances come from the radiology department of the "Klinikum für Strahlentherapie" in Rostock[1]. The instances correspond to radiotherapy plans for the treatment of prostate cancers. We use 25 different instances whose the size varies from 12x21 to 23x20. The size of the instances and the value of the maximal value L of the matrix (the minimal value is always equal to 0) are given in Table 9.3.

We first experiment the different rules for the adaptation of the heuristic of Engel (see section 9.3.1). We then expose the results obtained with 2PPLS.

9.4.2 Rules for the heuristic of Engel

We experiment here the different rules for the selection of the intervals in the heuristic of Engel. The rule "Min" is to take the first feasible interval which is the closest to the preceding interval. The rule "First" ("Last") is to take the first (last) feasible interval according to the order defined in section 9.3.1. The rule "Kali" is the rule developed by Kalinowski.

Instances of Engel

To compare the four rules, we first use the instances of Engel. For each value of L we make the average on 1000 different matrices for three values: the DC objective, the SU_{var} objective and the SU_{var} optimized objective which is the value of SU_{var} obtained after optimization of the sequence of segments with the LKH heuristic. For each matrix, as the heuristic is mainly deterministic (only the final optimization of SU_{var} with LKH is not), only one execution is run.

The results are given in Table 9.4 for DC and in Table 9.5 for SU_{var} and SU_{var} optimized.

We remark that for the DC objective, the "Kali" rule is better than the others. That is logical since the "Kali" rule is suited to minimize DC. And we have, except for $L = 4$, Kali $\succ$ Last $\succ$ First $\succeq$ Min.

[1] We thank Antje Keisel from the Institue for Mathematics of the University of Rostock to have provided us these instances.

Table 9.3: Size of the real instances.

Number	Size	L
1	12x21	28
2	23x20	30
3	16x20	28
4	23x20	25
5	23x20	35
6	22x20	25
7	15x21	31
8	13x21	31
9	23x20	20
10	23x20	18
11	13x20	25
12	14x21	31
13	13x21	25
14	13x21	26
15	20x20	16
16	23x19	18
17	23x20	17
18	22x19	27
19	22x20	15
20	14x20	29
21	13x19	30
22	13x20	13
23	13x20	26
24	23x19	13
25	23x19	13

Table 9.4: Average values of DC obtained by the different rules for the random instances (A=15x15).

	DC			
L	Min	First	Last	Kali
3	10.32	10.32	9.93	**9.72**
4	11.73	11.74	11.33	**10.94**
5	12.69	12.69	12.29	**11.76**
6	13.56	13.56	13.14	**12.50**
7	14.27	14.26	13.85	**13.12**
8	14.91	14.90	14.48	**13.71**
9	15.52	15.46	15.09	**14.20**
10	16.04	15.98	15.59	**14.69**
11	16.46	16.37	16.01	**15.06**
12	16.92	16.81	16.49	**15.46**
13	17.30	17.18	16.87	**15.81**
14	17.62	17.53	17.21	**16.13**
15	18.01	17.91	17.55	**16.52**
16	18.29	18.16	17.89	**16.79**

Table 9.5: Average values of SU_{var} and SU_{var} optimized obtained by the different rules for the random instances (A=15x15).

	SU_{var}				SU_{var} optimized			
L	Min	First	Last	Kali	Min	First	Last	Kali
3	79.69	**69.57**	70.38	113.68	77.43	63.05	**61.23**	103.14
4	94.22	**83.74**	84.10	127.63	91.72	74.72	**71.63**	114.44
5	104.81	96.55	**95.11**	137.43	101.82	84.85	**80.34**	122.28
6	112.35	107.29	**105.53**	146.62	108.98	93.19	**88.62**	129.35
7	119.49	117.02	**114.22**	153.55	115.75	101.14	**95.83**	134.90
8	126.53	126.12	**122.94**	160.69	122.39	108.12	**102.30**	140.40
9	133.32	133.65	**129.75**	166.67	128.38	114.32	**107.66**	144.95
10	139.10	141.08	**137.05**	172.46	134.04	120.17	**113.93**	149.60
11	144.16	147.09	**142.57**	177.08	138.56	125.22	**118.31**	152.92
12	149.41	153.17	**148.31**	182.08	143.47	130.06	**122.71**	156.71
13	153.56	158.54	**153.42**	186.16	147.34	134.47	**126.85**	160.10
14	**157.19**	162.91	157.96	189.64	150.70	138.10	**130.82**	162.75
15	**162.00**	168.27	163.02	194.47	154.91	142.46	**135.04**	166.35
16	**166.22**	172.34	167.41	197.64	158.84	146.25	**138.53**	169.18

On the other hand, if we want to minimize SU_{var}, we remark that the "Last" rule allows to obtain better results for the SU_{var} optimized objective. And we have Last $\succ$ First $\succ$ Min $\succ$ Kali.

If we do not consider the optimization step, the rule "First" is the best for $L = 3$ and $L = 4$, the "Last" rule is the best for L going from 5 to 13 and the rule "Min" is the best for L going from 14 to 16. The "Kali" rule is always the worst.

The "Last" rule seems thus better than the rule of Kalinowski for the SU_{var} objective, even if this rule gives higher DC value on average. The running time of the heuristic of Engel with each of these rules is negligible.

We do not stop the experiments here since in Figure 9.9, we show the evolution of SU_{var} optimized (average on 100 instances) according to L, going here from 10 to 10000 (in order to study the asymptotic behavior of the rules), for a 15x15 matrix. The "Last" and "Kali" rules are compared. We see on this figure that the difference between the "Last" and "Kali" rules is not so well marked than in Table 9.5. When L is bigger, there is anymore strong difference between both rules and in some cases, the "Kali" rule gives better results for the SU_{var} optimized objective.

We show in Figure 9.10 the evolution of SU_{var} optimized (average on 100 instances) according to the size of the matrix A. The value of L is fixed to 10. The matrix A is always considered as square and its size varies from 3x3 to 50x50. We see on this figure that here, the difference between the "Last" and "Kali" rules is more marked when the size of the matrix increases. The results given by the "Kali" rule are getting worse in comparison with the "Last" rule.

Real instances

The results for the 25 real instances considered are given in Tables 9.6 and 9.7.

For the DC value, we see that is most of the time the "Kali" rule that gives the best results but not always (on 5 instances, the "Last" rule gives a DC value lower than 1 or 2 units). For SU_{var} without optimization, on 19 instances the "Min" rule gives the best results. With optimization, the "Last" rule gives better results on 17 instances. The "Kali" rule never gives the best results.

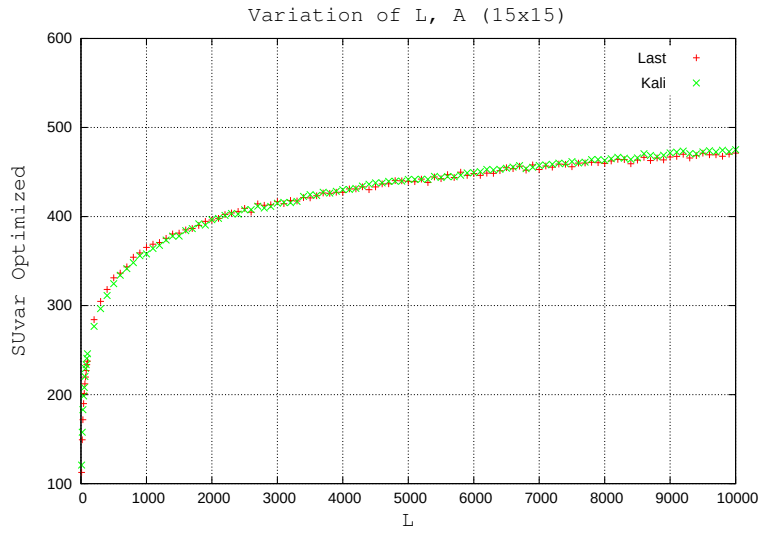

Figure 9.9: Evolution of SU_{var} according to L for a 15x15 matrix. The "Last" and "Kali" rules are compared.

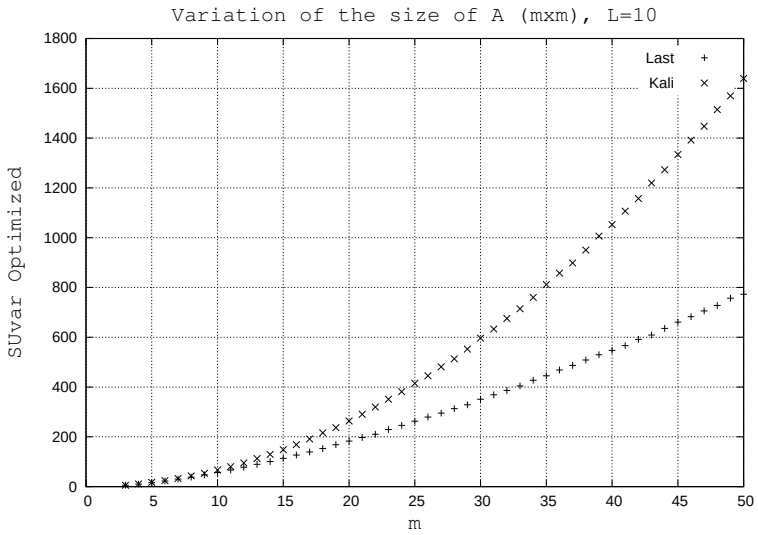

Figure 9.10: Evolution of SU_{var} according to the size of the square matrix m x m. The "Last" and "Kali" rules are compared.

Table 9.6: Values of DC obtained by the different rules for the real instances.

	DC			
Number	Min	First	Last	Kali
1	21	20	19	**18**
2	21	21	20	**18**
3	20	20	20	**19**
4	19	**18**	**18**	**18**
5	23	21	21	**20**
6	21	22	20	**19**
7	22	21	22	**20**
8	21	**20**	**20**	**20**
9	18	18	18	**17**
10	16	16	16	**14**
11	16	**15**	16	**15**
12	20	20	**19**	20
13	17	17	17	**16**
14	16	16	**13**	15
15	18	18	18	**16**
16	20	20	**18**	19
17	17	17	18	**16**
18	18	17	**16**	**16**
19	16	16	16	**15**
20	17	17	**16**	17
21	**16**	**16**	**16**	**16**
22	16	16	16	**15**
23	16	**15**	16	**15**
24	15	16	**14**	15
25	16	**15**	**15**	**15**

Table 9.7: Values of SU_{var} and SU_{var} optimized obtained by the different rules for the real instances.

	SU_{var}				SU_{var} optimized			
Number	Min	First	Last	Kali	Min	First	Last	Kali
1	**193**	229	218	256	177	195	**154**	208
2	**224**	226	228	254	216	207	**191**	225
3	**207**	238	216	266	199	194	**174**	229
4	241	243	**210**	275	239	235	**191**	260
5	**246**	257	247	289	241	231	**222**	270
6	**210**	246	243	288	208	214	**204**	273
7	**213**	225	239	263	205	197	**189**	215
8	202	**197**	217	255	191	**163**	193	216
9	**181**	225	223	256	**179**	210	184	233
10	**174**	230	204	224	**173**	202	179	189
11	157	169	**156**	192	153	148	**127**	148
12	**180**	193	217	278	177	**164**	179	235
13	**164**	175	170	199	157	**132**	142	165
14	152	168	**146**	180	151	151	**111**	145
15	**167**	**167**	173	228	158	143	**139**	198
16	**171**	175	197	265	171	162	**151**	231
17	**177**	180	180	211	165	**149**	**149**	181
18	187	176	**166**	205	186	160	**151**	179
19	**156**	173	160	188	155	144	**133**	165
20	**161**	173	190	221	161	158	**143**	184
21	**144**	157	166	198	141	147	**118**	167
22	**134**	155	165	191	131	139	**125**	153
23	**148**	161	174	181	**144**	**144**	148	154
24	**147**	**147**	164	214	143	**136**	137	199
25	143	**140**	159	194	141	**123**	125	159

9.4.3 Two-phase Pareto local search

We experiment in this section the 2PPLS method.

The initial population of 2PPLS is formed with two solutions: the first one is a good approximation of lexmin(DT,DC,SU_{var}) (obtained with the "Kali" rule and LKH) and the second one is a good approximation of lexmin(DT,SU_{var},DC) (obtained with the "Last" rule and LKH). If there is one solution that dominates the other, the initial population will be composed of only one solution: the non-dominated one.

For the neighborhood, we adopt the following choices:

- We try all possible segments for the segment that we put at the beginning of the new decomposition.
- Either we do not modify the segment or we modify it by trying all possibilities of modification. We modify each line of the segment separately, by considering all feasible possibilities for each line (equal to maximum 8): $\big((+1,+1),(+1,0),(+1,-1),(0,+1),(0,-1),(-1,+1),(-1,0),(-1,-1)\big)$.
- We try all feasible coefficients for the segment that we put at the beginning of the decomposition.
- The remaining matrix is decomposed with the heuristic of Engel, with the "Last" rule.

If the number of segments of the current decomposition is equal to K and the maximal value of the matrix equal to L, a crude bound for the number of neighbors generated is equal to $KL(8m+1)$.

As we explore entirely the neighborhood, the 2PPLS method is deterministic and will be applied only one time for each intensity matrix.

Instances of Engel

We first experiment the method on the random instances of Engel. As no state-of-the-art results are known for this multiobjective problem, we use very specific indicators to measure the quality of the approximations obtained.

In Table 9.8, we report the number of potentially efficient solutions found ($|PE|$), the number of phases of the PLS method before convergence and the running time in seconds (on a Intel Core 2 Duo T7500 2.2 GHz CPUs and 2 GB of RAM). We indicate for each indicator the minimum, maximum and mean values found. For each value of L we make the average on 20 different matrices.

We remark that:

- The number of potentially efficient solutions is not high: between 1 and 7, with a mean value between 1 and 4. The correlation between the objectives seems thus high for these instances.
- The number of phases is included between 1 and 16, with a mean value between 3 and 7, what means that the neighborhood is efficient since at each phase improvements are realized. Please remind that if there is no new potentially efficient solution generated during a phase, the PLS method stops since a Pareto local optimum set has been found. Moreover, we start PLS from relatively good initial solutions.
- The mean running time is acceptable, between 1 and 35 seconds. However, for some instances, the running time can be higher, until 124 seconds for an instance with $L = 15$.

The evaluation of the quality of the results is given in Table 9.9. We evaluate the improvements of DC by comparing the best values obtained with 2PPLS to the values obtained by the heuristic of Engel with the "Kali" rule (column "% DC - Kali"). We also evaluate the improvements of SU_{var} by comparing the best SU_{var} values obtained with 2PPLS to the SU_{var}

Table 9.8: Average values of the indicators obtained by 2PPLS for the random instances (1).

	$\|PE\|$			Number of phases			Time(s)		
L	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max
3	1.35	1	2	3.60	1	7	1.34	0.26	3.29
4	1.80	1	3	5.45	1	10	3.48	0.42	8.14
5	1.85	1	3	5.30	1	11	5.08	0.57	12.94
6	2.00	1	5	6.25	2	16	7.17	1.63	16.34
7	2.40	1	5	5.60	1	10	8.70	1.19	28.55
8	2.30	1	4	5.65	1	10	11.12	1.20	24.94
9	2.75	1	7	5.40	1	10	12.82	1.28	35.83
10	2.55	1	7	5.40	2	10	15.52	2.82	44.85
11	2.70	1	6	6.35	1	12	19.52	1.69	49.66
12	2.40	1	5	4.55	1	12	17.37	1.86	61.76
13	2.80	1	6	6.95	1	13	25.31	2.25	53.04
14	3.30	2	5	6.20	3	12	28.12	11.61	59.17
15	3.15	2	7	6.30	3	10	34.30	8.95	123.83
16	3.20	1	7	6.40	1	16	32.77	2.82	75.59

Table 9.9: Average values of the indicators obtained by 2PPLS for the random instances (2).

	DT optimal			DT not necessary optimal		
	% DC	% SU_{var}		% DC	% SU_{var}	
L	Kali	Kali+LKH	Last+LKH	Kali	Kali+LKH	Last+LKH
3	0.00	42.97	4.83	0.45	43.22	5.32
4	0.42	38.74	4.76	0.42	38.74	4.76
5	0.83	36.32	6.02	0.83	36.32	6.02
6	0.00	31.32	3.35	0.00	31.44	3.52
7	0.00	28.44	2.90	0.00	28.64	3.20
8	0.00	30.35	3.25	0.00	30.88	3.94
9	0.33	28.99	3.85	0.33	29.10	4.00
10	0.31	27.26	2.84	0.31	27.64	3.31
11	0.00	24.40	2.26	0.31	24.57	2.47
12	0.00	25.36	2.80	0.00	26.19	3.87
13	0.00	21.52	1.54	0.31	22.50	2.72
14	0.00	20.99	2.83	0.00	21.32	3.24
15	0.56	22.77	4.23	0.56	22.91	4.41
16	0.00	22.48	3.00	0.00	22.90	3.51

values obtained with the heuristic of Engel with the "Kali" rule and LKH (column "% SU_{var} - Kali+LKH") and to the SU_{var} values obtained with the heuristic of Engel with the "Last" rule and LKH (column "% SU_{var} - Last+LKH"). Two cases are distinguished: initially we evaluate the improvements made if we keep the optimal value for DT (column "DT optimal"), and secondly, we have no restriction on the value of the DT objective (column "DT not necessary optimal").

We see that the improvements of the DC values are very small. Indeed, the heuristic of Engel with the "Kali" rule is known to give near-optimal results for lexmin(DT,DC) on random instances [130]. On the other hand, the improvements of the SU_{var} values are remarkable. In comparison with the values obtained by the heuristic of Engel with the "Kali" rule and LKH, we obtain improvements from 20% to 43%. Comparing to the heuristic of Engel with the "Last" rule and LKH (which is one of the initial solution of 2PPLS), we obtain improvements from 1% to 6%.

We also see that allowing to deteriorate the DT value is only interesting for some instances. Both objectives DC and SU_{var} seem thus positively correlated with the DT objective.

Real instances

Similar results for the real instances are given in Tables 9.10 and 9.11.

We see that the number of potentially efficient solutions remains low: between 1 and 6. The number of phases is still interesting and is between 2 and 13. On the other hand, the running time is higher, between 3 and 1125 seconds. Indeed, the size of some of the instances is higher than the size of the random instances.

Table 9.10: Values of the indicators obtained by 2PPLS for the real instances (1).

Number	$\|PE\|$	Number of phases	Time(s)
1	2	9	39.71
2	4	8	310.53
3	5	13	235.31
4	1	4	40.74
5	6	10	1125.01
6	2	9	129.38
7	5	11	185.82
8	6	12	134.15
9	2	8	67.27
10	1	9	85.15
11	3	5	29.65
12	2	8	87.60
13	2	6	25.64
14	1	2	3.94
15	4	11	125.28
16	1	5	50.90
17	3	9	104.24
18	1	3	61.14
19	2	4	55.73
20	1	4	20.98
21	2	4	12.58
22	2	5	11.24
23	5	12	165.66
24	1	4	12.93
25	2	6	24.00

From the table 9.11, we see that we have improved the DC values comparing to the heuristic

of Engel for only 5 instances out of 25. On the other hand, for SU_{var}, comparing to the values obtained by the heuristic of Engel with the "Kali" rule and LKH, we obtain improvements from 13% to 37%. Comparing to the heuristic of Engel with the "Last" rule and LKH (which is one of the initial solution of 2PPLS), we obtain improvements from 0.65% to 15%. The DC and SU_{var} objectives are still positively correlated with the DT objective, since the results with DT not necessary optimal are very similar to those obtained if DT is optimal.

Table 9.11: Values of the indicators obtained by 2PPLS for the real instances (2).

	DT optimal			DT not necessary optimal		
	% DC	% SU_{var}		% DC	% SU_{var}	
Number	Kali	Kali+LKH	Last+LKH	Kali	Kali+LKH	Last+LKH
1	0.00	26.44	0.65	0.00	26.44	0.65
2	0.00	17.78	3.14	0.00	17.78	3.14
3	0.00	30.13	8.05	0.00	30.13	8.05
4	0.00	28.08	2.09	0.00	28.08	2.09
5	0.00	18.52	0.90	0.00	22.59	5.86
6	0.00	26.37	1.47	0.00	26.37	1.47
7	0.00	13.02	1.06	5.00	13.02	1.06
8	0.00	15.28	5.18	0.00	22.22	12.95
9	0.00	22.75	2.17	0.00	22.75	2.17
10	0.00	14.29	9.50	0.00	14.29	9.50
11	0.00	16.89	3.15	0.00	16.89	3.15
12	5.00	34.89	14.53	5.00	34.89	14.53
13	0.00	17.58	4.23	0.00	17.58	4.23
14	13.33	26.90	4.50	13.33	26.90	4.50
15	0.00	35.35	7.91	0.00	35.35	7.91
16	5.26	36.80	3.31	5.26	36.80	3.31
17	0.00	19.89	2.68	0.00	19.89	2.68
18	0.00	20.11	5.30	0.00	20.11	5.30
19	0.00	22.42	3.76	0.00	22.42	3.76
20	5.88	26.63	5.59	5.88	26.63	5.59
21	0.00	31.74	3.39	0.00	31.74	3.39
22	0.00	24.18	7.20	0.00	24.18	7.20
23	0.00	16.23	12.84	0.00	22.08	18.92
24	6.67	34.17	4.38	6.67	34.17	4.38
25	0.00	24.53	4.00	0.00	24.53	4.00

We give in Table 9.12 the values of the objectives for the six non-dominated points found for the real instance number 5 (of size 23x20). We see that is possible to reduce quite considerably the SU_{var} value if we do not use the solutions that give the optimal values for DT or DC.

Table 9.12: Values of the objectives for the six potentially non-dominated points found for the real instance number 5 (of size 23x20).

DT	DC	SU_{var}
91	20	260
91	21	220
92	20	241
92	21	212
93	22	211
97	22	209

9.5 Conclusion and perspectives

We have presented in this chapter first results for the problem of multiobjective decomposition of positive integer matrices, within the framework of the radiotherapy treatment. The results obtained with 2PPLS are encouraging since the method allows to improve state-of-the-art results.

More experimentations on different types of instances (size and characteristics) will have to be carried out to obtain more information about the correlation between the objectives and to validate the approach. Actually, we have already realized some experimentations to measure the correlation between the objectives, and we have obtained positive coefficients equal to 0.6 for (DT,DC) and (DC,SU_{var}) and a coefficient equal to 0.3 for (DT,SU_{var}). These results have been obtained by generating random solutions of random instances, but as the way the random instances and solutions are generated can have an impact on the values of the coefficients, the analysis should be made more thorough. These first results confirm however that the objectives are positively correlated.

For improving the solving method, we think that the key point will be to reduce the size of the neighborhood in order to be able to apply it not only from non-dominated solutions, as done in PLS. Indeed, in PLS, we do not apply too much the neighborhood since we only start it from new non-dominated solutions and this number remains low for all the instances treated. If it was not the case, the running time would be very high.

A method based on weight sets would be interesting only if the size of the neighborhood is reduced. Indeed, the weight sets would define different directions of research, and for each direction a local search method could be applied. That means that for each weight set, the neighborhood would be necessary. The running time of such a method will be thus high if the neighborhood is very time-consuming.

To reduce the size of the neighborhood, it would be necessary to deeply study what are the successful moves, what gives improvements of the solution and what never gives improvements. With such results, we will be able to reduce the size of the neighborhood and making it faster and consequently more efficient. Random choices in the neighborhood with a limited number of neighbors would also be a valuable option.

Defining a crossover operator would be interesting, and would allow to integrate the local search in an evolutionary algorithm.

We leave all these ideas for further research.

Finally, we have only considered the third problem that occurs in the IMRT optimization process, without taking into account the geometry and intensity problems (we suppose that the intensity matrices are given). Solving the three problems simultaneously would be very challenging (but as said by Ehrgott *et al.* in their recent survey [59]: “This is, however, beyond the capabilities of current optimization algorithms and computer power” !!).

10
Conclusion and perspectives

In this thesis, we have presented new methods based on metaheuristics for solving multiobjective combinatorial optimization (MOCO) problems. These methods have been applied to two classical MOCO problems: the multiobjective multidimensional knapsack problem (MOMKP) and the multiobjective traveling salesman problem (MOTSP). The optimization of a multiobjective problem that occurs in intensity-modulated radiotherapy treatment (IMRT) has also been considered.

After introducing the thesis in **chapter 1**, we have exposed in **chapter 2** the difficulties related to MOCO problems, that is essentially: the $\mathcal{NP}$-Hardness of these problems even when the single-objective counterpart is in $\mathcal{P}$, an exponential number of efficient solutions, and the presence of non-supported efficient solutions, harder to compute than supported ones.

Thereof, in **chapter 3**, we have seen that exact methods are not easy to apply and can only be used in order to solve small size MOCO problems. We have thus focused this chapter to the presentation of the different adaptations of local search and population-based metaheuristics to multiobjective optimization. The hybridation between local search and genetic algorithms, giving the memetic algorithms, has also been reviewed. The problem of assessing the quality of the results obtained by multiobjective metaheuristics (MOMHs), an essential point of this work since the results obtained by our methods are compared to state-of-the-art results, has been tackled in this chapter. For this point, we have presented the different unary indicators and tools that we use in this work to assess the quality of the approximations generated.

In **chapters 4 and 5**, state-of-the-arts of the methods dedicated to the resolution of respectively the MOMKP and the MOTSP have been realized. In these chapters, we have seen that no exact method has been developed with the aim of specifically solving these problems. Most of the approaches are thus based on metaheuristics, where hybrid algorithms, by mainly combining population-based method with local search, seem to give the best results.

In **chapter 6**, three new solving methods have been presented. The first one is a simple and classic adaptation to MOO of tabu search, called PRTS, with a selection of the neighbors based on Pareto dominance relations. A memetic algorithm, called MEMOTS, making use of PRTS, has been then exposed. The particularity of this algorithm is in the selection of the parents which is based on the density information given by a hypergrid built in the objective space, while most of the other memetic algorithms use scalarizing functions to pick out the parents. The last method, the two-phase Pareto local search (2PPLS), is an extension of the Pareto local search (PLS) method [6, 185], where in the place of one single initial solution, a population of solutions of good quality is used as starting point. We have proposed for the generation of the initial population, a heuristic adaptation of the first phase of the exact two-phase method, that is the dichotomic method of Aneja and Nair [4] allowing to generate the supported efficient solutions.

In **chapter 7**, we have applied the three methods in order to solve the MOMKP. The

stand-alone tabu search turned out to be rapidly inefficient, we have focused this chapter to the adaptation of MEMOTS and 2PPLS. Both methods displayed excellent results on biobjective instances. We have also realized the hybridization between MEMOTS and 2PPLS, in an elegant way since MEMOTS is used as subroutine in 2PPLS in order to explore efficiently the neighborhood of a solution. The role of MEMOTS is to rapidly generate potentially efficient solutions of small size instances, solutions that are then used to produce the neighbors. This hybridization gave a very competitive method, since we arrived to obtain better results on all indicators considered in this work, in reasonable computational times, in comparison with the indicators obtained by merging the results obtained by many other MOMHs. On three-objective instances, the performances are not so impressive but remain better than state-of-the-art algorithms for the indicators considered.

In **chapter 8**, we have dealt with the solving of the biobjective TSP (bTSP). We have only adapted 2PPLS, mainly because MEMOTS requires many parameters in comparison with 2PPLS which simply needs an initial population and a neighborhood. We have used simple components to adapt 2PPLS: an initial population generated with the heuristic adaptation of the Aneja and Nair method coupled with the Lin and Kernighan (LK) heuristic, and the two-edge exchange as neighborhood. In spite of this simplicity, we have obtained better results for many different types of instances than state-of-the-art results for the indicators considered, except for the clustered instances. We have also shown that the data perturbation technique can be used in order to improve the results of 2PPLS. We have then presented how to solve biobjective instances of a size never tackled before, that is 750 and 1000 cities. To this end, different speed-up techniques based on dominance relations performed in the data or based on the solutions generated during the first phase, have been developed. These speed-up techniques are very efficient since they allow to greatly reduce the running time of 2PPLS without deteriorating the quality of the results.

Chapter 9 is devoted to the application that we have opted to tackle in this work. In this very specific problem that occurs in IMRT optimization, the aim is to decompose a positive integer matrix into a set of segments, binary segments, having to respect the consecutive ones property. Three criteria have been considered for assessing the quality of a decomposition: the sum of the coefficients of the decomposition, the cardinality of the decomposition and the overall travel time, related to the sum of distances between two consecutive segments. This is the first time that this multiobjective problem has been tried to be solved. We have first proved that the objectives were conflicting through an example. We have then applied the 2PPLS method, in order to generate an approximation of the non-dominated points of this problem. With a large neighborhood and an approximation of the global lexicographic optimal solutions as initial population, we have obtained promising results for random and real instances. Even if the objectives are generally conflicting, we have noticed that few non-dominated points exist for this problem, which, however does not call into question the use of 2PPLS.

If we come back now to the problem exposed in the introduction, we can wonder if, after this long study, we have helped the engineer of the NASA to find efficient planets!

Fortunately, what we can do with computers and fictive problems cannot as easily be done in real life and therefore, we think that, unfortunately, the engineer will feel alone and worthless when he will be thrown out in space with so many planets to explore...

Indeed, as the tabu search developed in this work, in which only one solution is evolving in the search space, the engineer in his spaceship can follow only one path. We have shown that MEMOTS is efficient but with this method, we need to frequently jump in different regions of the search space, which is something hardly achievable for the engineer. In 2PPLS, a *population* of solutions is evolving, each solution exploring locally a part of the search space. Moreover, the size of this population is not fixed at the beginning and can increase during the search.

Therefore, in order to improve the quality of the list of potentially efficient planets found by the engineer, we assume that the physicists should work on the two following problems:

- To study teleportation in order to jump very rapidly from one point to another in space.
- To study how to launch a set of spaceships into space, with each ship being able to generate new ships.

In addition to investigating these two problems, to realize a study of the search space before the expedition would be worthwhile, in order to reduce the space to explore. That could be done, as realized with the speed-up techniques applied to the bTSP, with telescopes allowing to see *inside* the planets, in order to identify what are the essential components of a good planet (for example, a planet with a mean temperature higher than a certain threshold has few chances to be efficient). To know if the efficient planets are connected would also be relevant, or at least to know if efficient planets are often close to other efficient planets.

Physicists are now well-equipped to take an informed decision in order to help the engineer!

Finally, we believe that our contributions are more than having generated new state-of-the-art results for the MOMKP, bTSP or the radiotherapy application. With 2PPLS, we hope that other classic MOCO problems could be solved, by combining high quality single-objective solvers and the PLS method. As for most classic CO problems, there exist very efficient single-objective heuristics, there is no reason not to use them in order to tackle the MO case. Once an efficient single-objective solver is known, the questions that remain are how to use the solver, which technique to use for generating the weight sets in the first phase, when stopping the first phase and which neighborhood to use. The neighborhood is applied intensively, it thus does not have to be too large, but large enough to allow to connect potentially efficient solutions and to sufficiently explore the search space.

We mention here that recently, Depuydt in his master's thesis under the direction of Tuyttens and ourself [47], has applied 2PPLS in order to solve the biobjective set covering problem. He obtained better results on several indicators than state-of-the-art results.

Even if a thesis is a long work, we often have to limit ourself, causing in the same time the setting aside of many other interesting subjects. We enumerate here the main perspectives related to this work (some have already been mentioned in previous chapters):

1. **PLS for single-objective optimization**

   The PLS method can be seen in a single-objective context, as a record-to-record [50] metaheuristic coupled with a population. The record-to-record is a deterministic local search, where a neighbor is accepted if its evaluation is less than the RECORD value plus a deviation D. The value of RECORD corresponds to the best evaluation of the visited solutions during the search. As the RECORD value can only decrease with time, the criterion of selection of a neighbor is more and more tight with time too. Therefore, in order to adapt PLS to single-objective optimization, to evaluate a solution, it is enough to take as first objective the value of the objective to optimize, and as second objective, the gap between the RECORD value and the value of the first objective. Accordingly, the solution presenting the RECORD value has a second objective equal to zero. The difference with the classic record-to-record metaheuristic is that we explore the neighborhood of all solutions checking the criterion of acceptance, by managing a population, with an unlimited size.

   This interpretation can explain why, in the case of the MOTSP, the results obtained by the LK heuristic on weighted sums are sometimes improved by PLS that uses simple two-edge exchange moves. Indeed, we did not mention it before, but we have compared the qualities of the solutions obtained by 2PPLS and the qualities of the solutions generated at the end of the first phase (solutions generated with the LK heuristic), on weighted sums. In order to do that, we have used 101 uniformly distributed weighted sets, and we have measured the number of times that the solutions of 2PPLS were better than the solutions of the first phase. As in the first phase, we optimize weighted sums with an efficient heuristic, we could expect no improvement with simple two-edge exchange moves

realized in PLS. However, we improve, on average on all the instances, in 21% of the cases the values obtained by the solutions generated at the end of the first phase, for the 101 weighted sums. Therefore, with simple two-edge exchange moves, we managed to improve the results obtained by the LK heuristic, which uses more complex moves. This result reflects well the "record-to-record" behavior of PLS: PLS works autonomously in many directions, and a solution good for one weighted sum is an interesting starting point for the optimization of another weighted sum. It would be thus interesting to test PLS on single-objective CO problems.

2. **First phase of 2PPLS**

   As we have seen in this work (especially for the bTSP), the quality of the solutions generated during the first phase has a strong impact on the quality of the results of 2PPLS. Ideally, during the first phase, at least one solution of each connected components (or clusters) of the adjacency graph (see section 2.2.3 in chapter 2) built on the base of the neighborhood used in PLS, should be generated. In order to achieve this goal, we could improve the results of the first phase, by adding the data perturbation technique, as done with the MOTSP, but also with crossover operators. These techniques should allow to produce solutions located in different connected components of the preceding solutions, according to the neighborhood considered in PLS. We have also seen than using the supported efficient solutions as solutions in the first phase is not always more efficient than using an approximation of these supported efficient solutions. The initial population should be thus enough diversified in order to have initial solutions in every regions of the search space. Another issue is when to stop the first phase, if a given amount of computational time is given.

3. **Pareto global search**

   The 2PPLS method stops when a local optimum (defined in the MO case) has been found, which is very basic in comparison with single-objective metaheuristics. Therefore, integrating techniques allowing to escape from this local optimum could be integrated. This idea has already been mentioned in the thesis of Paquete [183]. For instance, we can realize that by perturbing the local optimum in order to start the search from another region of the search space, like in the iterated local search. Another option is, as in the variable neighborhood search metaheuristic, to use several neighborhoods, and applying a larger neighborhood once a local optimum has been found. It can also be achieved by simply accepting dominated neighbors in the population with a certain probability.

4. **Coupling 2PPLS with exact methods**

   In the adaptation of 2PPLS to the MOMKP, we have coupled 2PPLS with an exact subroutine to solve small instances of the problem. However, this subroutine was a simple exact method based on enumeration, which turned out to be ineffective. Therefore, exacts methods for the MOMKP or the MOTSP could be very helpful for solving heuristically, but in a very efficient way, instances of a size that the stand-alone exact method will not be able to solve in a reasonable time.

5. **Speed-up techniques**

   We only applied speed-up techniques to the MOTSP. However, these techniques can be applied to other classic MOCO problems, like the MOMKP. Indeed, in his master's thesis in 2006, Bouchez [25] has shown that for several MOCO problems efficient solutions only use a small part of the data set. Reducing the search space can do but nothing than strongly decreasing the running time of MOMHs. In the case of the MOTSP, we have heuristically reduced the search space, but it would be interesting to also reduce it exactly, that is by guaranteeing that the reduced search space still contains all efficient solutions. Also, a pre-processing phase could be interesting to apply in order to rapidly obtain an overview of the promising regions of the search space, by for example analyzing the efficient solutions

of relaxed version of the problems treated (for example, for the MOTSP, that can be done by analyzing the edges used by the spanning tree relaxation and for the MOMKP, by analyzing the items used by the linear relaxation).

However, speed-up techniques would be more difficult to apply in the case where the objectives are not of the same type, like in the radiotherapy application.

6. **New methods**

   Since an efficient set contains many solutions, it seems appropriate to work with methods that create a statistical model of the potentially efficient set, which would be based on the information contained in the solutions. With this model, new solutions could be easily produced. We refer to the estimation of distribution algorithms [146] or the cross-entropy method [199]. For example, for the MOTSP, we did not use at all the frequencies with which the edges are used in the solutions generated during the first phase.

7. **New problems - New instances**

   Many MOCO problems have not yet been studied or only briefly. For the MOMKP, only random instances have been tackled. For the MOTSP, to our knowledge, the asymmetric version has never been studied, and only some of the variants (see section 5.4 in chapter 5). We have also only dealt with the biobjective version of the MOTSP, and at least the three-objective case should be studied. Howevever, the case where the number of objectives is higher to three often requires different solving techniques that the ones exposed in this work [72].

8. **Approximation algorithms**

   Few approximation algorithms have been developed for solving MOCO problems, and only for particular problems. It would be very interesting, but a priori not trivial, to develop a 2PPLS method with guarantee of performance, by first generating the initial population with the dichotomic method of Aneja and Nair coupled with approximation algorithms and then evaluating the improvements that can generate the application of PLS.

9. **Dealing with many solutions**

   We have seen that even with biobjective problems, the number of non-dominated points can be very high. Therefore, integrating techniques in MEMOTS or 2PPLS in order to reduce the number of non-dominated points can be relevant in some cases. With this aim, the hypergrid can be used to divide the objective space and to retain only one non-dominated points in each division of the hypergrid. Note that this technique is not new and is used for example in the M-PAES method of Knowles and Corne [138].

10. **Interactive methods**

    The obvious issue of our approach to solve MOCO problems is that we generate a (very) large set of solutions, while the decision maker (DM) is only interested in *one* solution, "good" in regard of his preferences. Therefore, generating so many solutions can cause computational overhead and difficulties for the DM to select one solution. Moreover, in the case of large number of objectives, the algorithms do not where to go since everything seems promising.

    As we have shown that the methods developed in this thesis are effective for solving the MOCO problems considered, it would be interesting to adapt these methods in order to generate a small set of solutions corresponding to the preferences of the DM. A promising way to deal with that has been introduced recently by Auger *et al.* [8, 9] with the weighted hypervolume, which is an indicator in which arbitrary user preferences can be articulated, and therefore allows to direct the search according to the DM preferences.

Bibliography

[1] F. Ben Abdelaziz, J. Chaouachi, and S. Krichen. A hybrid heuristic for multiobjective knapsack problems. In S. Voss, S. Martello, I. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 205–212. Kluwer Academic Publishers, Dordrecht, 1999.

[2] F. Ben Abdelaziz and S. Krichen. A tabu search heuristic for multiobjective knapsack problems. Technical Report RRR 28-97, Rutgers Center for Operations Research, Piscataway, USA, 1997.

[3] M.J. Alves and M. Almeida. MOTGA: A multiobjective Tchebycheff based genetic algorithm for the multidimensional knapsack problem. *Computers & Operations Research*, 34:3458–3470, 2007.

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

[5] E. Angel, E. Bampis, and L. Gourves. Approximating the Pareto curve with local search for the bicriteria TSP(1,2) problem. *Theoretical Computer Science*, 310:135–146, 2004.

[6] E. Angel, E. Bampis, and L. Gourvès. A dynasearch neighborhood for the bicriteria traveling salesman problem. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, pages 153–176. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535, Berlin, 2004.

[7] D. Applegate. Chained Lin-Kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, 15:82–92, 2003.

[8] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Articulating User Preferences in Many-Objective Problems by Sampling the Weighted Hypervolume. In G. Raidl et al., editors, *Genetic and Evolutionary Computation Conference (GECCO 2009)*, pages 555–562, New York, NY, USA, 2009. ACM.

[9] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler. Investigating and Exploiting the Bias of the Weighted Hypervolume to Articulate User Preferences. In G. Raidl et al., editors, *Genetic and Evolutionary Computation Conference (GECCO 2009)*, pages 563–570, New York, NY, USA, 2009. ACM.

[10] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Potasi. *Complexity and Approximation - Combinatorial Optimization Problems and Their Approximability Properties*. Springer Verlag, 1999.

[11] D. Baatar, N. Boland, S. Brand, and P.J. Stuckey. Minimum cardinality matrix decomposition into consecutive-ones matrices: CP and IP approaches. In *CPAIOR '07: Proceedings of the 4th international conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 1–15, Berlin, Heidelberg, 2007. Springer-Verlag.

[12] D. Baatar, H. Hamacher, M. Ehrgott, and G. Woeginger. Decomposition of integer matrices and multileaf collimator sequencing. *Discr Appl Math*, 152:6–34, 2005.

[13] E. Balas and E. Zemel. An algorithm for large zero-one knapsack problems. *Operational Research*, 28:1130–1154, 1980.

[14] V. Barichard and J-K. Hao. Genetic tabu search for the multi-objective knapsack problem. *Journal of Tsinghua Science and Technology*, 8(1):8–13, 2003.

[15] V. Barichard and J.K. Hao. An empirical study of tabu search for the MOKP. In *Proceedings of the First International Workshop on Heuristics*, volume 4, pages 47–56, China, 2002. Series of Information & Management Sciences.

[16] M. Basseur. Design of cooperative algorithms for multi-objective optimization: application to the flow-shop scheduling problem. *4OR*, 4(3):255–258, 2006.

[17] C. Bazgan, H. Hugot, and D. Vanderpooten. Implementing an efficient FPTAS for the 0-1 multi-objective knapsack problem. *European Journal of Operational Research*, 198(1):47–56, 2009.

[18] C. Bazgan, H. Hugot, and D. Vanderpooten. Solving efficiently the 0-1 multi-objective knapsack problem. *Computers & Operations Research*, 36(1):260–279, 2009.

[19] R.P. Beausoleil, G. Baldoquin, and R.A. Montejo. Multi-start and path relinking methods to deal with multiobjective knapsack problems. *Annals of Operations Research*, 157:105–133, 2008.

[20] J.L. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, 4:387–411, 1992.

[21] N. Beume, C.M. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold. On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13:1075–1082, 2009.

[22] C. Blum, M.J. Blesa Aguilera, A. Roli, and M. Sampels, editors. *Hybrid Metaheuristics, An Emerging Approach to Optimization*, volume 114 of *Studies in Computational Intelligence*. Springer, 2008.

[23] P.C. Borges and M.P. Hansen. A study of global convexity for a multiple objective travelling salesman problem. In C.C. Ribeiro and P. Hansen, editors, *Essays and surveys in metaheuristics*, pages 129–150. Kluwer, 2000.

[24] T. Bortfeld, A. Boyer, D. Kahler, and T. Waldron. X-ray filed compensation with multileaf collimators. *Int J Radiat Oncol Biol Phys*, 28(3):723–730, 1994.

[25] F-X. Bouchez. Résolution de problèmes d'optimisation combinatoire multi-objectif à l'aide du logiciel d'optimisation CPLEX. Master's thesis, Faculté Polytechnique de Mons, 2006.

[26] V.J. Bowman. On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. In H. Thiriez and S. Zionts, editors, *Multiple criteria decision making*, pages 76–86. Springer. Lecture Notes in Economics and Mathematical Systems, Berlin, 1976.

[27] K. Bringman and T. Friedrich. Approximating the least hypervolume contributor: $\mathcal{NP}$-hard in general, but fast in practice. In Ehrgott et al. [54], pages 6–20.

[28] R. Burkard. Open problem session. In *Conference on Combinatorial Optimization*, pages 24–29, Oberwolfach, November 2002.

[29] M. E. Captivo, J.C.N. Clímaco, J.R. Figueira, E.Q.V. Martins, and J. L. Santos. Solving bicriteria 0-1 knapsack problems using a labeling algorithm. *Computers & Operations Research*, 30(12):1865–1886, 2003.

[30] B. Codenotti, G. Manzini, L. Margara, and G. Resta. Perturbation: An efficient technique for the solution of very large instance of the euclidean TSP. *INFORMS Journal on Computing*, 8:125–133, 1996.

[31] C.A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, editors. *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005, Proceedings*, volume 3410 of *Lecture Notes in Computer Science*. Springer, 2005.

[32] C.A. Coello Coello, D.A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002.

[33] J.L. Cohon. *Multiobjective Programming and Planning*. Academic Press, New York, 1978.

[34] D. Conforti, F. Guerriero, and R. Guido. Optimization models for radiotherapy patient scheduling. *4OR*, 6(3):263–278, 2008.

[35] P. Czyzak and A. Jaszkiewicz. Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7:34–47, 1998.

[36] C. Gomes da Silva, J. Clímaco, and J. R. Figueira. Scatter search method for the bi-criteria multi-dimensional {0,1}-knapsack problem using surrogate relaxation. *Journal of Mathematical Modelling and Algorithms*, 3(3):183–208, 2004.

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

[38] C. Gomes da Silva, J. Clímaco, and J. R. Figueira. Core problems in bi-criteria {0,1}-knapsack problems. *Computers & Operations Research*, 35(1):2292–2306, 2008.

[39] C. Gomes da Silva, J. R. Figueira, and J. Clímaco. Integrating partial optimization with scatter search for solving bi-criteria {0,1}-knapsack problems. *European Journal of Operational Research*, 177:1656–1677, 2007.

[40] G. Dantzig. Discrete variable extremum problems. *Operations Research*, 5:226–227, 1957.

[41] K. Dächert and J. Gorski. Numerical studies of Tchebycheff-type approaches for solving multicriteria optimization problems. Presented at the EURO XXIII conference, 2009.

[42] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, New-York, 2001.

[43] K. Deb. A robust evolutionary framework for multi-objective optimization. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 633–640, New York, NY, USA, 2008. ACM.

[44] K. Deb and T. Goel. Controlled elitist non-dominated sorting genetic algorithms for better convergence. In Zitzler et al. [250], pages 67–81.

[45] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 42–50, Washington, 1989. Springer. Lecture Notes in Computer Science No. 1917.

[46] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[47] M. Depuydt. Analyse et résolution via la métheuristique 2PPLS de problèmes de couverture biobjectif. Master's thesis, Faculté Polytechnique de Mons, 2009.

[48] M. Dorigo, A. Colorni, and V. Maniezzo. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1991.

[49] M. Dorigo and T. Stützle. *Ant Colony Optimization.* The MIT Press, Cambridge, 2004.

[50] G. Dueck. New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104(1):86–92, 1993.

[51] M. Dyer. Calculating surrogate constraints. *Mathematical Programming*, 19:255–278, 1980.

[52] M. Dyer, A. Frieze, R. Kannan, A. Kapoor, L. Perkovic, and U. Vazirani. A mildly exponential time algorithm for approximating the number of solutions to a multidimensional knapsack problem. *Combinatorics, Probability and Computing*, 2:271–284, 1993.

[53] M. Ehrgott. *Multicriteria Optimization. Second edition.* Springer, Berlin, 2005.

[54] M. Ehrgott, C.M. Fonseca, X. Gandibleux, J-K. Hao, and M. Sevaux, editors. *Evolutionary Multi-Criterion Optimization, 5th International Conference, EMO 2009, Nantes, France, April 7-10, 2009. Proceedings*, volume 5467 of *Lecture Notes in Computer Science*. Springer, 2009.

[55] M. Ehrgott and X. Gandibleux. Approximation algorithms for combinatorial multicriteria optimization problems. *International Transactions in Operational Research*, 7:5–31, 2000.

[56] M. Ehrgott and X. Gandibleux. Multiobjective combinatorial optimization. In M. Ehrgott and X. Gandibleux, editors, *Multiple Criteria Optimization – State of the Art Annotated Bibliographic Surveys*, volume 52, pages 369–444. Kluwer Academic Publishers, Boston, MA, 2002.

[57] M. Ehrgott and X. Gandibleux. Bound sets for biobjective combinatorial optimization problems. *Computers & Operations Research*, 34:2674–2694, 2007.

[58] M. Ehrgott and X. Gandibleux. Hybrid metaheuristics for multi-objective combinatorial optimization. In C. Blum, M. J. Blesa Aguilera, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 114 of *Studies in Computational Intelligence*, pages 221–259. Springer, 2008.

[59] M. Ehrgott, Ç. Güler, H. W. Hamacher, and L. Shao. Mathematical optimization in intensity modulated radiation therapy. *4OR: a Quarterly Journal of Operations Research*, 6(3):199–262, 2008.

[60] M. Ehrgott, H. Hamacher, and M. Nußbaum. Decomposition of matrices and static multileaf collimators: a survey. In C. Alves, P. Pardalos, and L. Vincente, editors, *Optimization in medicine*, pages 27–48. Springer, Berlin, 2007.

[61] M. Ehrgott and K. Klamroth. Connectedness of efficient solutions in multiple criteria combinatorial optimization. *European Journal of Operational Research*, 97:159–166, 1997.

[62] S. Elaoud, T. Loukil, and J. Teghem. The Pareto fitness genetic algorithm: Test function study. *European Journal of Operational Research*, 177(3):1703–1719, 2007.

[63] S. Elaoud, J. Teghem, and T. Loukil. Multiple crossover genetic algorithm for the multiobjective traveling salesman problem: University of Sfax (Tunisia). *Submitted for publication*, 2008.

[64] V. Emelichev and V. Perepelitsa. On cardinality of the set of alternatives in discrete many-criterion problems. *Discrete Mathematics and Applications*, 2(5):461–471, 1992.

[65] K. Engel. A new algorithm for optimal multileaf collimator field segmentation. *Discrete Appl. Math.*, 152(1-3):35–51, 2005.

[66] C. Engelbeen and S. Fiorini. Constrained decompositions of integer matrices and their applications to intensity modulated radiation therapy. *To appear in Networks*, 2009.

[67] T. Erlebach, H. Kellerer, and U. Pferschy. Approximating multiobjective knapsack problems. *Management Science*, 48(12):1603–1612, 2002.

[68] T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

[69] T.S. Ferguson. *Mathematical Statistics, a decision theoretic approach*. Academic Press, New-York and London, 1967.

[70] J.R. Figuera, M. Wiecek, and G. Tavares. Multiple criteria knapsack problems : Network models and computational results. In *Proceedings of the multi-Objective Programming and Goal Programming Conference (MOPGP'06)*, Tours, 2006.

[71] R. Fisher and K. Richter. Solving a multiobjective traveling salesman problem by dynamic programming. *Mathematische Operationsforschung und Statistik, Series Optimization*, 13(2):247–252, 1982.

[72] P.J. Fleming, R.C. Purshouse, and R.J. Lygoe. Many-objective optimization: An engineering design perspective. In Coello et al. [31], pages 14–32.

[73] C.M. Fonseca and P.J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[74] C.M. Fonseca, P.J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors. *Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003, Faro, Portugal, April 8-11, 2003, Proceedings*, volume 2632 of *Lecture Notes in Computer Science*. Springer, 2003.

[75] X. Gandibleux and A. Freville. Tabu Search Based Procedure for Solving the 0-1 Multi-Objective Knapsack Problem: The Two Objectives Case. *Journal of Heuristics*, 6(3):361–383, August 2000.

[76] X. Gandibleux and K. Klamroth. Cardinality bounds for multiobjective knapsack problems based on weighted sums scalarizations. In *Proceedings of the multi-Objective Programming and Goal Programming Conference (MOPGP'06)*, Tours, 2006.

[77] X. Gandibleux, N. Mezdaoui, and A. Fréville. A Tabu Search Procedure to Solve Combinatorial Optimisation Problems. In R. Caballero, F. Ruiz, and R.E. Steuer, editors, *Advances in Multiple Objective and Goal Programming*, volume 455 of *Lecture Notes in Economics and Mathematical Systems*, pages 291–300. Springer-Verlag, 1997.

[78] X. Gandibleux, H. Morita, and N. Katoh. The supported solutions used as a genetic information in a population heuristics. In Zitzler et al. [250], pages 429–442.

[79] C. Garcia-Martinez, O. Cordon, and F. Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research*, 180(1):116 – 148, June 2007.

[80] M.R. Garey and D.S. Johnson. *Computers and intractibility. A guide to the theory of NP-completeness*. Freeman and Company, New York, 1979.

[81] M. Gendreau, J-F. Bérubé, and J-Y. Potvin. An exact epsilon-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. *European Journal of Operational Research*, 194(1):39–50, 2009.

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

[83] F. Glover. A multiphase dual algorithm for the zero-one integer programming problem. *Operations Research*, 13(6):879–919, 1965.

[84] F. Glover. Surrogate constraint duality in mathematical programming. *Operations Research*, 23:434–451, 1975.

[85] F. Glover. Optimization by ghost image processes in neural networks. *Computers & Operations Research*, 21(8):801–822, 1994.

[86] F. Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65(1-3):223–253, 1996.

[87] F. Glover. Scatter search and path relinking. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 297–316. McGraw Hill, London, 1999.

[88] F. Glover. Multi-start and strategic oscillation methods-principles to exploit adaptive memory. In M. Laguna and J.L. Gonzalez-Velarde, editors, *Computing tools for modeling, optimization and simulation: interfaces in computer science and operations research*, pages 1–24. Kluwer Academic, Dordrecht, 2000.

[89] F. Glover and G. Kochenberger. *Handbook of Metaheuristics*. Kluwer, Boston, 2003.

[90] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic, Dordrecht, 1997.

[91] D.E. Goldberg. Genetic algorithms for search, optimization, and machine learning. *Readin, MA: Addison-Wesley*, 1989.

[92] J. Gorski. Analysis of the connectedness of Pareto-optimal solutions in multiple criteria combinatorial optimization. Master's thesis, University of Erlangen-Nuremberg, 2004.

[93] J. Gorski, K.Klamroth, and S. Ruzika. Connectedness of efficient solutions in multiple objective combinatorial optimization. ISSN 1435-5833 310, Universität Erlangen-Nürnberg, 2006.

[94] V. Grunert da Fonseca, C.M. Fonseca, and A.O. Hall. Inferential Performance Assessment of Stochastic Optimisers and the Attainment Function. In E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 213–225. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[95] A. Gupta and A. Warburton. Approximation methods for multiple criteria traveling salesman problems, towards interactive and intelligent decision support systems. In Y. Sawaragi, editor, *Proc. 7th Internat. Conf. on Multiple Criteria Decision Making*, pages 211–217, Berlin, 1986. Springer.

[96] G. Gutin and A. Punnen. *The Traveling Salesman Problem and its Variations*. Kluwer, Dordrecht / Boston / London, 2002.

[97] Y. Haimes, L. Lasdon, and D. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1:296–297, 1971.

[98] H.W. Hamacher and G. Ruhe. On spanning tree problems with multiple objectives. *Annals of Operations Research*, 52:209–230, 1994.

[99] M.P. Hansen. Use of Substitute Scalarizing Functions to Guide a Local Search Based Heuristic: The Case of moTSP. *Journal of Heuristics*, 6(3):419–430, 2000.

[100] M.P. Hansen and A. Jaszkiewicz. Evaluating the quality of approximations of the non-dominated set. Technical report, Technical University of Denmark, Lingby, Denmark, 1998.

[101] P. Hansen. Bicriterion path problems. *Lecture Notes in Economics and Mathematical Systems*, 177:109–127, 1979.

[102] P. Hansen and N. Mladenovic. First vs. best improvement: An empirical study. *Discrete Appl. Math.*, 154:802–817, 2006.

[103] K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126:106–130, 2000.

[104] J.H. Holland. *Adaptation in natural and artificial systems*. PhD thesis, University of Michigan Press, 1975.

[105] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

[106] H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Elsevier, 2004.

[107] B. Huang, L. Yao, and K. Raguraman. Bi-level GA and GIS for multi-objective TSP route planning. *Transportation Planning and Technology*, 29(2):105–124, April 2006.

[108] H. Isermann. The enumeration of the set of all efficient solutions for a linear multiple objective program. *Operations Research Quarterly*, 28:711–725, 1977.

[109] H. Ishibuchi and T. Murada. A multi-objective genetic local search algorithm and its application to flow shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 28(3):392–403, 1998.

[110] H. Ishibuchi and T. Murata. Multi-Objective Genetic Local Search Algorithm. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation*, pages 119–124, Nagoya, Japan, 1996. IEEE.

[111] H. Ishibuchi and K. Narukawa. Recombination of Similar Parents in EMO Algorithms. In Coello et al. [31], pages 265–279.

[112] A. Jaszkiewicz. Genetic Local Search for Multiple Objective Combinatorial Optimization. Technical Report RA-014/98, Institute of Computing Science, Poznań University of Technology, 1998.

[113] A. Jaszkiewicz. Experiments done with the MOMHLIB: http://www-idss.cs.put.poznan.pl/ jaszkiewicz/momhlib/. Technical report, Institute of Computing Science, Poznań University of Technology, 2000.

[114] A. Jaszkiewicz. On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem—A Comparative Experiment. Technical Report RA-002/2000, Institute of Computing Science, Poznań University of Technology, Poznań, Poland, July 2000.

[115] A. Jaszkiewicz. A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the Pareto memetic algorithm. Technical Report RA-003/01, Institute of Computing Science, Poznań University of Technology, Poznań, Poland, 2001.

[116] A. Jaszkiewicz. Comparison of local search-based metaheuristics on the multiple-objective knapsack problem. *Foundations of Computing and Decision Sciences*, 26(1):99–120, 2001.

[117] A. Jaszkiewicz. Multiple objective metaheuristic algorithms for combinatorial optimization. Habilitation thesis, 2001.

[118] A. Jaszkiewicz. Genetic Local Search for Multiple Objective Combinatorial Optimization. *European Journal of Operational Research*, 137(1):50–71, 2002.

[119] A. Jaszkiewicz. On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem—A Comparative Experiment. *IEEE Transactions on Evolutionary Computation*, 6(4):402–412, August 2002.

[120] A. Jaszkiewicz. On the Computational Efficiency of Multiple Objective Metaheuristics. The Knapsack Problem Case Study. *European Journal of Operational Research*, 158(2):418–433, October 2004.

[121] A. Jaszkiewicz and P. Zielniewicz. Pareto memetic algorithm with path-relinking for biobjective traveling salesman problem. *European Journal of Operational Research*, 193(3):885–890, 2009.

[122] D.S. Johnson, G. Gutin, L.A. McGeoch, A.Yeo, W. Zhang, and A. Zverovich. Experimental analysis of heuristics for the ATSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer, 2002.

[123] D.S. Johnson and L.A. McGeoch. The Traveling Salesman Problem: A Case Study in Local Optimization. In E. H. L. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley and Sons Ltd, 1997.

[124] D.S. Johnson and L.A. McGeoch. Experimental analysis of heuristics for the STSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer, 2002.

[125] F. Jolai, M.J. Rezaee, M. Rabbani, J. Razmi, and P. Fattahi. Exact algorithm for biobjective 0-1 knapsack problem. *Applied Mathematics and Computation*, 194(2):544–551, 2007.

[126] D.F. Jones, S.K. Mirrazavi, and M. Tamiz. Multi-objective meta-heuristics: an overview of the current state-of-art. *European Journal of Operational Research*, 137(1):1–9, 2002.

[127] J. Jorge and X. Gandibleux. Nouvelles propositions pour la résolution exacte du problème de sac à dos bi-objectif unidimensionnel en variables binaires, février 2007. FRANCORO V – MOSIM'07.

[128] N. Jozefowiez, F. Glover, and M. Laguna. Multi-objective meta-heuristics for the traveling salesman problem with profits. *Journal of Mathematical Modelling and Algorithms*, 7(2):177–195, 2008.

[129] N. Jozefowiez, F. Semet, and E-G. Talbi. Multi-objective vehicle routing problems. *European Journal of European Research*, 189(2):293–309, 2008.

[130] T. Kalinowski. Realization of intensity modulated radiation fields using multileaf collimators. In *General Theory of Information Transfer and Combinatorics*, pages 1010–1055. Springer Berlin / Heidelberg, 2006.

[131] T. Kapamara, K. Sheibani, O.C.L Haas, C.R. Reeves, and D. Petrovic. A review of scheduling problems in radiotherapy. In Coventry University Publishing, editor, *Proceedings of the International Control Systems Engineering Conference (ICSE 2006)*, Coventry (U.K), 2006.

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

[133] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Berlin, 2004.

[134] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[135] K. Klamroth and M. Wiecek. Dynamic programming approaches to the multiple criteria knapsack problem. *Naval Research Logistics*, 47(1):57–76, 2000.

[136] J. Knowles and D. Corne. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation. In *1999 Congress on Evolutionary Computation*, pages 98–105, Washington, D.C., July 1999. IEEE Service Center.

[137] J. Knowles and D. Corne. A Comparison of Diverse Approaches to Memetic Multiobjective Combinatorial Optimization. In *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*, pages 103–108, Las Vegas, Nevada, July 2000.

[138] J. Knowles and D. Corne. M-PAES: A Memetic Algorithm for Multiobjective Optimization. In *2000 Congress on Evolutionary Computation*, volume 1, pages 325–332, Piscataway, New Jersey, July 2000. IEEE Service Center.

[139] J. Knowles and D. Corne. Memetic algorithms for multiobjective optimization: issues, methods and prospects. In N. Krasnogor, J.E. Smith, and W.E. Hart, editors, *Recent Advances in Memetic Algorithms*, pages 313–352. Springer, 2004.

[140] J. Knowles, L. Thiele, and E. Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, February 2006.

[141] J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, February 1956.

[142] K. Kumar, A.H. Joshi, K.K. Banka, and P.I. Rockett. Evolution of hyperheuristics for the biobjective 0/1 knapsack problem by multiobjective genetic programming. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1227–1234, New York, NY, USA, 2008. ACM.

[143] R. Kumar and N. Banerjee. Analysis of a multiobjective evolutionary algorithm on the 0-1 knapsack problem. *Theoretical Computer Science*, 358(1):104–120, 2006.

[144] R. Kumar and P.K. Singh. Pareto evolutionary algorithm hybridized with local search for biobjective TSP. In C. Grosan, A. Abraham, and H. Ishibuchi, editors, *Hybrid Evolutionary Algorithms*, chapter 14. Springer-Verlag, 2007.

[145] P. Larranaga, C.M.H. Kuijpers, R.H. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170, 1999.

[146] P. Larranaga and J.A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation (Genetic Algorithms and Evolutionary Computation).* Springer, October 2001.

[147] M. Laumanns, E. Zitzler, and L. Thiele. On the Effects of Archiving, Elitism, and Density Based Selection in Evolutionary Multi-objective Optimization. In Zitzler et al. [250], pages 181–196.

[148] H. Lee and P.S. Pulat. Bicriteria network flow problems: Integer case. *European Journal of Operational Research*, 66(1):148–157, April 1993.

[149] H. Li, Q. Zhang, E. Tsang, and J.A. Ford. Hybrid estimation of distribution algorithm for multiobjective knapsack problem. In *Evolutionary Computation in Combinatorial Optimization*, pages 145–154. Springer, Berlin / Heidelberg, 2000.

[150] W. Li. Finding Pareto-optimal set by merging attractors for a bi-objective traveling salesman problem. In Coello et al. [31], pages 797–810.

[151] A. Liefooghe, L. Jourdan, and E-G. Talbi. A Unified Model for Evolutionary Multiobjective Optimization and its Implementation in a General Purpose Software Framework: ParadisEO-MOEO. Research Report RR-6906, INRIA, 2009.

[152] S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21:498–516, 1973.

[153] G. Lizárraga, A. Hernández Aguirre, and S.B. Rionda. On the Possibility to Create a Compatible-Complete Unary Comparison Method for Evolutionary Multiobjective Algorithms. In *2008 Genetic and Evolutionary Computation Conference (GECCO'2008)*, pages 759–760, Atlanta, USA, July 2008. ACM Press. ISBN 978-1-60558-131-6.

[154] M. López-Ibáñez and T. Stützle. An analysis of algorithmic components for multiobjective ant colony optimization: A case study on the biobjective TSP. Technical Report TR/IRIDIA/2009-01, IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium, 2009.

[155] T. Loukil, J. Teghem, and P. Fortemps. Solving multi-objective production scheduling problems with Tabu Search. *Control and Cybernetics*, 29(3):819–828, 2000.

[156] T. Loukil, J. Teghem, and D. Tuyttens. Solving multiple objective production scheduling problems using metaheuristics. *European Journal of Operational Research*, 161(1):42–61, May 2005.

[157] H.R. Lourenço, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics, volume 57 of International Series in Operations Research and Management Science*, pages 321–353. Kluwer Academic Publishers, Norwell, MA, 2002.

[158] T. Lust. Mise au point, implémentation et comparaison de métaheuristiques pour l'optimisation combinatoire multicritère. Master's thesis, Faculté Polytechnique de Mons, 2005.

[159] T. Lust and A. Jaszkiewicz. Speed-up techniques for solving large-scale biobjective TSP. *Computers & Operations Research*, 37:521–533, 2010.

[160] T. Lust and J. Teghem. Multicriteria maintenance problem resolved by tabu search. In *Proceedings of the 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM'2006)*, pages 1–6, Saint-Etienne, May 2006.

[161] T. Lust and J. Teghem. PRTS+D et MEMOTS : Nouvelles métaheuristiques pour l'optimisation combinatoire multicritère. In *Actes des articles longs sélectionnés lors du 7ème congrès de la ROADEF*, pages 137–151, Lille, February 2006. Presses Universitaires de Valenciennes.

[162] T. Lust and J. Teghem. Memots: a memetic algorithm integrating tabu search for combinatorial multiobjective optimization. *RAIRO: Operations Research*, 42(1):3–33, 2008.

[163] T. Lust and J. Teghem. Multiobjective decomposition of positive integer matrix: Application to radiotherapy. In Ehrgott et al. [54], pages 335–349.

[164] T. Lust and J. Teghem. The multiobjective traveling salesman problem: a survey and a new approach. *To appear as a chapter for the Springer book on advances in multi-objective nature inspired computing*, 2009.

[165] T. Lust and J. Teghem. Two-phase Pareto local search for the biobjective traveling salesman problem. *To appear in Journal of Heuristics*, 2009.

[166] H.B. Mann and D.R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18:50–60, 1947.

[167] B. Manthey and L. Shankar Ram. Approximation algorithms for multi-criteria traveling salesman problems. *CoRR*, arxiv:abs/cs/0606040, 2007.

[168] S. Martello and P. Toth. *Knapsack Problems*. Wiley, New York, 1990.

[169] E.Q.V. Martins and J.L.E. Dos Santos. The labeling algorithm for the multiobjective shortest path problem. Technical Report 99/005 CISUC, Departamento de Matemática, Universidade de Coimbra, November 1999.

[170] I.I. Melamed and K.I. Sigal. The linear convolution of criteria in the bicriteria traveling salesman problem. *Computational Mathematics and Mathematical Physics*, 37(8):902–905, 1997.

[171] P. Merz and B. Freisleben. Memetic algorithms for the traveling salesman problem. *Complex Systems*, 13:297–345, 2001.

[172] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[173] Z. Michalewicz and J. Arabas. Genetic algorithms for the 0/1 knapsack problem. In Z.W. Ras and M. Zemankova, editors, *Methodologies for Intelligent Systems Conference (ISMIS)*, pages 134–143, Berlin, 1994.

[174] K. Miettinen. *Nonlinear multiobjective optimization*. Kluwer, Boston, 1999.

[175] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[176] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Technical Report C3P 826, Caltech Concurrent Computation Program, 1989.

[177] S. Mostaghim and J. Teich. Quad-trees: A Data Structure for Storing Pareto Sets in Multiobjective Evolutionary Algorithms with Elitism. In A. Abraham, L. Jain, and R. Goldberg, editors, *Evolutionary Multiobjective Optimization: Theoretical Advances And Applications*, pages 81–104. Springer-Verlag, London, 2005.

[178] T. Murata and H. Ishibuchi. MOGA: Multi-Objective Genetic Algorithms. In *Proceedings of the 2nd IEEE International Conference on Evolutionary Computing*, pages 289–294, Perth, Australia, November 1995.

[179] R.A. Novelline. *Squire's Fundamentals of Radiology*. Harvard University Press, 1997.

[180] O. Ozpeynirci. *Approaches for multiobjective combinatorial optimization problems*. PhD thesis, University of Economics, Department of Logistics Management, Izmir, Turkey, 2008.

[181] C. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithm and complexity*. Englewood Cliffs: Prentice-Hall, 1982.

[182] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources (extended abstract). In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 86–92, 2000.

[183] L. Paquete. *Stochastic Local Search Algorithms for Multiobjective Combinatorial Optimization: Methods and Analysis*. PhD thesis, FB Informatik, TU Darmstadt, 2005.

[184] L. Paquete, M. Chiarandini, and T. Stützle. Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, pages 177–199, Berlin, 2004. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535.

[185] L. Paquete, M.Chiarandini, and T.Stützle. A study of local optima in the multiobjective travelling salesman problem. Technical Report AIDA-02-07, FG Intellektik, FB Informatik, TU Darmstadt, Germany, 2002. Presented at the Multiobjective Metaheuristics Workshop (MOMH 2002), Paris, 4-5 November, 2002.

[186] L. Paquete, T. Schiavinotto, and T. Stützle. On local optima in multiobjective combinatorial optimization problems. *Annals of Operations Research*, 156(1):83–97, 2007.

[187] L. Paquete and T. Stützle. A Two-Phase Local Search for the Biobjective Traveling Salesman Problem. In Fonseca et al. [74], pages 479–493.

[188] L. Paquete and T. Stützle. Clusters of non-dominated solutions in multiobjective combinatorial optimization. In V. Barichard, M. Ehrgott, X. Gandibleux, and V. T'kindt, editors, *Multiobjective Programming and Goal Programming: Theoretical Results and Practical Applications*, pages 69–77. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 618, Berlin, 2009.

[189] L. Paquete and T. Stützle. Design and analysis of stochastic local search for the multiobjective traveling salesman problem. *Computers & Operations Research*, 36(9):2619–2631, 2009.

[190] D. Payne. Intensity Modulated Radiation Therapy. *In touch*, pages 1–2, 2003.

[191] M. Pelikan, K. Sastry, and E. Cantu-Paz. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications (Studies in Computational Intelligence)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[192] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401, 1957.

[193] A. Przybylski, X. Gandibleux, and M. Ehrgott. Two-phase algorithms for the biobjective assignement problem. *European Journal of Operational Research*, 185(2):509–533, 2008.

[194] A. Przybylski, X. Gandibleux, and M. Ehrgott. Two recursive algorithms for finding all nondominated extreme points in the outcome set of multi-objective integer programme. *INFORMS Journal on Computing, in press*, 21(4), 2009.

[195] A. Raith and M. Ehrgott. A two-phase algorithm for the biobjective integer minimum cost flow problem. *Computers & Operations Research*, 36(6):1945–1954, 2009.

[196] R.M. Ramos, S. Alonso, J. Sicilia, and C. Gonzalez. The problem of the optimal biobjective spanning tree. *European Journal of Operational Research*, 111:617–628, 1998.

[197] C. Rego and F. Glover. Local search and metaheuristics for the traveling salesman problem. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*. Kluwer, 2002.

[198] G. Reinelt. TSPLIB - a traveling salesman problem library. *ORSA Journal of Computing*, 3(4):376–384, 1991.

[199] R.Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99:89–112, 1997.

[200] G. Ruhe and B. Fruhwirth. ϵ-optimality for bicriteria programs and its application to minimum cost flows. *Computing*, 44:21–34, 1990.

[201] H.M. Safer and J.B. Orlin. Fast approximation schemes for multi-criteria combinatorial optimization. Technical Report 3756-95, Sloan School of Management, MIT, Cambridge, MA, 1995.

[202] H.M. Safer and J.B. Orlin. Fast approximation schemes for multi-criteria flow, knapsack and scheduling problems. Technical Report 3757-95, Sloan School of Management, MIT, Cambridge, MA, 1995.

[203] F. Samanlioglu, W.G. Ferrell Jr., and M.E. Kurz. A hybrid random-key genetic algorithm for a symmetric travelling salesman problem. *Computers and Industrial Engineering*, 55(2):439–449, 2008.

[204] H. Sato, H.E. Aguirre, and K. Tanaka. Local dominance and local recombination in MOEAs on 0/1 multiobjective knapsack problems. *European Journal of Operational Research*, 181(3):1708–1723, 2007.

[205] J.D. Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.

[206] W. Schlegel and A. Mahr. *3D-conformal radiation therapy: a multimedia introduction to methods and techniques*. Springer, Heidelberg, Berlin, 2002.

[207] H. Schmitz and S. Niemann. A bicriteria traveling salesman problem with sequence priorities. In K. Sörensen, M. Sevaux, W. Habenicht, and M.J. Geiger, editors, *Metaheuristics in the Service Industry*, pages 1–14. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 624, Berlin, 2009.

[208] P. Serafini. Some considerations about computational complexity for multiobjective combinatorial problems. In J. Jahn and W. Krabs, editors, *Recent advances and historical development of vector optimization, volume 294 of LNEMS*, pages 222–232. Springer-Verlag, Berlin, 1986.

[209] P. Serafini. Simulated annealing for multi-objective optimization problems. In *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making*, volume 1, pages 87–92, Taipei, 1992.

[210] R.A.C. Siochi. Minimizing static intensity modulation delivery time using an intensity solid paradigm. *Int J Radiat Oncol Biol Phys*, 43:671–680, 1999.

[211] R.S. Solanki, P.A. Appino, and J.L. Cohon. Approximating the noninferior set in multiobjective linear programming problems. *European Journal of Operational Research*, 68:356–373, 1993.

[212] N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.

[213] K. Steiglitz and P. Weiner. Some improved algorithms for computer solution of the traveling salesman problem. In *Proceedings 6th Ann. Allerton Conf. on Communication, Control, and Computing*, pages 814–821, Urbana, 1968. Department of Electrical Engineering and the Coordinated Science Laboratory, University of Illinois.

[214] R. Steuer. *Multiple Criteria Optimization: Theory, Computation and Applications*. John Wiley & Sons, New-York, 1986.

[215] M. Sun and R.E. Steuer. Quad-trees and linear lists for identifying nondominated criterion vectors. *INFORMS Journal on Computing*, 8(4):367–375, 1996.

[216] Z. Taşkin, J. Smith, H. Romeijn, and J. Dempsey. Optimal multileaf collimator leaf sequencing in IMRT treatment planning. Technical report, Department of Industrial and Systems Engineering, University of Florida, 2007.

[217] E-G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5):541–564, 2002.

[218] E-G. Talbi. *Metaheuristics: From Design to Implementation.* Wiley-Blackwell, 2009.

[219] J. Teghem. *Programmation linéaire. Deuxième édition.* Editions de l'Université de Bruxelles, Bruxelles, 2003.

[220] J. Teghem. La programmation linéaire multicritère. In D. Dubois and M. Pirlot, editors, *Concepts et méthodes pour l'aide à la décision*, pages 215–288. Hermès, 2006.

[221] J. Teghem and P. Kunsch. A survey of techniques for finding efficient solutions to multi-objective integer linear programming. *Asia-Pacific Journal of Operational Research*, 3(2):95–108, 1986.

[222] J. Teghem and M. Pirlot. *Optimisation approchée en recherche opérationnelle.* Hermès science, 2002.

[223] J. Teghem, D. Tuyttens, and E.L. Ulungu. An interactive heuristic method for multi-objective combinatorial optimization. *Computer & Operations Research*, 27:621–634, 2000.

[224] C.T. Tung. A multicriteria Pareto-optimal algorithm for the traveling salesman problem. *Asia-Pacific Journal of Operational Research*, 11:103–115, 1994.

[225] D. Tuyttens. Private communication, 2006.

[226] D. Tuyttens, J. Teghem, and N. El-Sherbeny. A Particular Multiobjective Vehicle Routing Problem Solved by Simulated Annealing. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, pages 133–152. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535, Berlin, 2004.

[227] D. Tuyttens, J. Teghem, Ph. Fortemps, and K. Van Nieuwenhuyze. Performance of the MOSA Method for the Bicriteria Assignment Problem. *Journal of Heuristics*, 6(3):295–310, August 2000.

[228] E.L. Ulungu. *Optimisation Combinatoire multicritère: Détermination de l'ensemble des solutions efficaces et méthodes interactives.* PhD thesis, Université de Mons-Hainaut, Faculté des Sciences, Mons, Belgique, 1993.

[229] E.L. Ulungu and J. Teghem. Heuristic for multiobjective combinatorial optimization problems with simulated annealing. Presented at the EURO XII conference, 1992.

[230] E.L. Ulungu and J. Teghem. Multiobjective combinatorial optimization problems: A survey. *Journal of Multi-Criteria Decision Analysis*, 3:83–104, 1994.

[231] E.L. Ulungu and J. Teghem. The two-phases method: An efficient procedure to solve biobjective combinatorial optimization problems. *Foundation of Computing and Decision Science*, 20:149–156, 1995.

[232] E.L. Ulungu, J. Teghem, and Ph. Fortemps. Heuristics for multi-objective combinatorial optimization by simulated annealing. In J. Gu, G. Chen, Q. Wei, and S. Wang, editors, *Multiple Criteria Decision Making: Theory and Applications. Proceedings of the 6th National Conference on Multiple Criteria Decision Making*, pages 228–238, Windsor, UK, 1995. Sci-Tech.

[233] E.L. Ulungu, J. Teghem, Ph. Fortemps, and D. Tuyttens. MOSA Method: A Tool for Solving Multiobjective Combinatorial Optimization Problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236, 1999.

[234] E.L. Ulungu, J. Teghem, and Ch. Ost. Efficiency of interactive multi-objective simulated annealing through a case study. *Journal of the Operational Research Society*, 49:1044–1050, 1998.

[235] D. Vandensteen, F. Hulstaert, and C. Camberlin. Radiothérapie conformationnelle avec modulation d'intensité (IMRT). Technical Report KCE reports vol. 62B, Centre fédéral d'expertise des soins de santé (Bruxelles), 2007.

[236] V.V. Vazirani. *Approximation Algorithms*. Springer Verlag, 2001.

[237] A. Viana and J. Pinho de Sousa. Using metaheuristics in multiobjective resource constrained project scheduling. *European Journal of Operational Research*, 120(2):359–374, January 2000.

[238] D.S. Vianna and J.E.C. Arroyo. A GRASP algorithm for the multi-objective knapsack problem. In *QEST '04: Proceedings of the The Quantitative Evaluation of Systems, First International Conference*, pages 69–75, Washington, DC, USA, 2004. IEEE Computer Society.

[239] P. Vincke. *L'aide multicritère à la décision*. Editions de l'Université de Bruxelles, Bruxelles, 1989.

[240] M. Visée, J. Teghem, M. Pirlot, and E.L. Ulungu. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization*, 12:139–155, 1998.

[241] G. M. G. H. Wake, N. Boland, and L. S. Jennings. Mixed integer programming approaches to exact minimization of total treatment time in cancer radiotherapy using multileaf collimators. *Computers & Operations Research*, 36(3):795–810, 2009.

[242] A. Warburton. Approximation of Pareto optima in multiple-objective shortest-path problems. *Operations Research*, 35(1):70–79, 1987.

[243] S. Webb. Intensity-modulated radiation therapy (IMRT): a clinical reality for cancer treatment, "any fool can understand this". The 2004 Silvanus Thompson Memorial Lecture. *British Institute of Radiology*, 78:64–72, 2005.

[244] L. While, L. Bradstreet, L. Barone, and P. Hingston. Heuristics for optimising the calculation of hypervolume for multi-objective optimisation problems. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, Edinburgh, UK, 2005.

[245] L.A. Wolsey and G.L. Nemhauser. *Integer and Combinatorial Optimization*. Wiley-Interscience, November 1999.

[246] Z. Yan, L. Zhang, L. Kang, and G. Lin. A new MOEA for multiobjective TSP and its convergence property analysis. In Fonseca et al. [74], pages 342–354.

[247] Q. Yang and S. Ding. Novel algorithm to calculate hypervolume indicator of Pareto approximation set. *CoRR*, abs/0704.1196, 2007.

[248] C.W. Zhang and H.L. Ong. Solving the biobjective zero-one knapsack problem by an efficient LP-based heuristic. *European Journal of Operational Research*, 159(3):545–557, 2004.

[249] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

[250] E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, and D. Corne, editors. *Evolutionary Multi-Criterion Optimization, First International Conference, EMO 2001, Zurich, Switzerland, March 7-9, 2001, Proceedings*, volume 1993 of *Lecture Notes in Computer Science*. Springer, 2001.

[251] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

[252] E. Zitzler, M. Laumanns, L. Thiele, C.M. Fonseca, and V. Grunert da Fonseca. Why Quality Assessment of Multiobjective Optimizers Is Difficult. In W.B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 666–673, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[253] E. Zitzler and L. Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 1998.

[254] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.

[255] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V. Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.

[256] P. Zygmanski, F. Hacker, S. Friesen, R. Rodenbush, H-M. Lu, and L. Chin. Maximum MLC opening effect in dynamic delivery of IMRT: leaf-positional analysis. *Journal of applied clinical medical physics*, 6(2), 2005.